CROATICA CHEMICA ACTA
CCACAA **81** (2) 347–350 (2008)

ISSN-0011-1643
CCA-3251
Conference Paper

An Algorithm to Enumerate a Special Class of Digraphs: Application to Water Clusters*

Damir Vukičević[a,] and Ante Graovac[b,c]**

[a]*Faculty of Science, Department of Mathematics, University of Split, Nikole Tesle 12, HR-21000 Split, Croatia*

[b]*Faculty of Science, Department of Chemistry, University of Split, Nikole Tesle 12, HR-21000 Split, Croatia*

[c]*The Ruđer Bošković Institute, P. O. Box 180, HR-10002 Zagreb, Croatia*

RECEIVED NOVEMBER 8, 2007; REVISED FEBRUARY 14, 2008; ACCEPTED FEBRUARY 18, 2008

Keywords
digraphs
recursive algorithms
enumeration
optimization of algorithms
water clusters
hydrogen bonding

Recently Miyake and Aida have developed a directed graph model and related algorithm which generates all possible topologically distinct hydrogen-bonded clusters of water molecules. Here we present a new algorithm based on recursive functions. Numerical results show that our algorithm is much faster than that of Miyake and Aida.

INTRODUCTION

The study of water clusters is continuously in the focus of scientific attention. Computational tasks to analyze stability and dynamics of such clusters are demanding.[1–4] However, prior to any computation one has to enumerate all possible topologically distinct clusters. Here, the discrete mathematics plays an important role. One possible discrete model is to represent water clusters by a special class of directed graphs, *i.e.* digraphs,[5] where vertices correspond to water molecules and arcs (directed bonds) to hydrogen bonds from proton-donor of one molecule to proton-acceptor of another water molecule. The procedure, which we further call the MA algorithm, to obtain all possible topologically distinct clusters has been developed by Miyake and Aida[1,2] which have been able to enumerate all possible clusters with up to eight water molecules. As the number of topologically distinct clusters grows enormously with the number of water molecules in the cluster, it is of interest to develop an efficient graph algorithm[6] to enumerate distinct clusters. Here, we present an algorithm which is much faster than the MA one as it is based on the recursive enumeration.

The underlying connected graph G representing a cluster of n water molecules is a graph on n vertices with some number, m, of edges, where an edge describes a presence of hydrogen bond between two water molecules. If, for each edge, we know which of its terminal vertices (water molecules) donates a proton and which one accepts this proton, then instead of G, we describe the totality of directed bonding by a digraph H. As a wa-

* This paper is dedicated to the memory of Professor Dean Rosenzweig.

** Author to whom correspondence should be addressed. (E-mail: vukicevi@pmfst.hr)

ter molecule can donate mostly up to two protons to other water molecules and accept mostly up to two protons, in corresponding digraph H, there are mostly up to two outgoing and mostly up to two ingoing arcs, *i.e.* the valency of each vertex in underlying graph G is mostly up to four. Further, a simultaneous donation and acceptance of protons along a given bonds in H is not allowed, *i.e.* there are no undirected bond in H.

Adjacency matrix is convenient way to represent connectivity in the graph:[7–9] for a graph G with n vertices, the adjacency matrix $\boldsymbol{A} = \boldsymbol{A}(\mathrm{G})$ is the n-th order square matrix whose element a_{ij} equals 1 if vertices i and j are connected by an edge in G, and 0 otherwise. For a digraph H the element a_{ij} is 1 if there is an arc going from vertex i to vertex j, and 0 otherwise.

For *e.g.* clusters of three water molecules there are only two distinct undirected graphs: 3-cycle and a path of length 3. The underlying graph of the path cluster is shown in Figure 1a together with its adjacency matrix. There are three topologically distinct digraphs of path and one of them is shown with its adjacency matrix in Figure 1b. A digraph shown together with its adjacency matrix in Figure 1c is equivalent to one shown in Figure 1b.

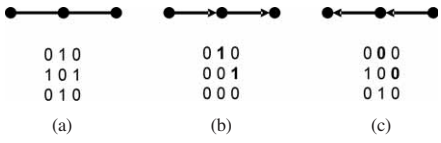

Figure 1. Underlying graph and two equivalent digraphs of path cluster of three water molecules with their adjacency matrices.

METHOD

The MA method considers all possible adjacency matrices of underlying graphs. It proceeds further by eliminating graphs having vertices of degrees larger than 4, while this restriction is implemented as a core of our algorithm. However, both methods have to eliminate disconnected graphs and duplicates.

After having at disposal the underlying graphs, for each of them one has to consider all its corresponding digraphs satisfying the above restrictions on valencies. The MA method considers all possible orientations of edges and then eliminates those that don't satisfy restrictions on valencies, where we adhere to the same restrictions, but we put them in the very core of our recursive algorithm.

The last step in both methods is to eliminate duplicates. Here, we introduce the notion of canonical labeling. Let consider the adjacency matrix of graph shown in Figure 1a. Its non-zero elements above the main diagonal are marked as bolded. The corresponding elements in adjacency matrices of two isomorphic digraphs shown in Figures 1b and 1c read as binary numbers are 11 and 00, respectively. The digraph with smaller, in general with the smallest among such numbers, we call canonically labeled digraph. Our method eliminates duplicates much quicker than the MA method, because we consider only the automorphisms of the underlying graph instead of considering all the permutations of vertices. Namely, as for a graph on n vertices there are $n!$ permutations to be considered in the MA method, in our method one considers only a few automorphisms.

ALGORITHM

We use the following variables in our algorithm (and we assume that at the beginning of the execution of the program all variables are set to zero):

• *NumVertices* is the number of vertices of the graphs under consideration. It is the input given to the program (it is assumed that *NumVertices* ≥ 7 as the results of enumeration for the graphs with lower number of vertices have been already given by Miyake and Aida[1,2]);

• *MaxDeg* is the maximum degree of the graph considered. Three options are successively examined: *MaxDeg* = 2,3,4;

• *a* is the matrix in which the adjacency matrix of a graph (respectively digraph) G (respectively H) will be stored;

• *deg* (*indeg, outdeg*) are arrays in which degrees (indegrees, outdegrees) of each vertex are stored;

• *NumOfGraphs* (*NumOfDigraphs*) stores the number of the graphs (digraphs) with the required properties;

• *NumOfAutomorphisms* stores the number of automorphisms of the observed graph;

• *Aut* stores all the automorphisms of the observed graph;

• *NumAut* is the number of the automorphisms of the observed graph;

• *NumEdges* stores the number of edges;

• *Edges* is *NumEdges* × 2 array in which edges are stores. *Edges* [*i*][1] and *Edges* [*i*][2] are vertices that are incident to the *i*-th edge (the first and the second vertex uniquely determnine the edge).

The algorithm (in the pseudo-code) is given below:

```
main( )
For MaxDeg = 2,...,4
    deg[1] = MaxDeg
    For i = 2,..., MaxDeg + 1
        a[1][i] = 1,a[i][1] = 1
        deg[i] = 1
    For i = MaxDeg + 2,...NumVertices
        a[1][i] = 0,a[i][1] = 0
        deg[i] = 0
```

```
        rec(2,3) // this function is described below
END

rec(x,y)
If y = NumVertices + 1
    If x = NumVertices
        If graph G is connected
            If IsGraphCanonical() // this function
              checks if graph has the form described
              in the paper[1] and we skip its pseudocode
            Analyze() // this function is described below
    Else
        rec(x+1,x+2)
Else
    If NumVertices + 1 – y + deg[x] ≥ 2
        rec(x,y+1)
    If deg[x] < MaxDeg and deg[y] < MaxDeg
        a[x][y] = 1, a[y][x] = 1, deg[x] = deg[x]+1,
          deg[y] = deg[y]+1
        rec(x,y+1)
        a[x][y] = 0, a[y][x] = 0, deg[x] = deg[x]–1,
          deg[y] = deg[y]–1
END

Analyze( )
NumOfGraphs = NumOfGraphs + 1
NumOfAutomorphisms = 0
FindAllAutomorphisms // this is a standard graph-
  theoretical algorithm, hence we skip its pseudocode
NumEdges = 0
For 1 ≤ i < j ≤ n
    If a[i][j] = 1
        NumEdges = NumEdges + 1,
          edges[NumEdges][1] = i,
          edges[NumEdges][2] = j
For i = 1,...,n
    InDeg[i] = 0, OutDeg[i] = 0
RecOrient(1,2) // this function is described below
END

RecOrient(x,y)
If y = NumVertices
    If x = NumVertices
        If IsDigraphCanonical( ) // this function is
          described below
            NumOfDigraphs = NumOfDigraphs + 1
        Else
            RecOrient(x+1,x+2)
Else If a[x][y] = 0
    RecOrient(x,y+1)
Else
    If (OutDeg[x] < 2) and (InDeg[y] < 2)
        a[x][y] = 2, a[y][x] = 0,
          OutDeg[x] = OutDeg[x] + 1,
          InDeg[y] = InDeg[y]+1
        RecOrient(x,y+1)
        a[x][y] = 1, a[y][x] = 1,
          OutDeg[x] = OutDeg[x] – 1,
          InDeg[y] = InDeg[y] – 1
    If (InDeg[x] < 2) and (OutDeg[y] < 2)
        a[x][y] = 0, a[y][x] = 2,
          InDeg[x] = InDeg[x] + 1,
          OutDeg[y] = OutDeg[y] + 1
        RecOrient(x,y + 1)
        a[x][y] = 1, a[y][x] = 1,
          InDeg[x] = InDeg[x] – 1,
          OutDeg[y] = OutDeg[y] – 1
END

IsDigraphCanonical()
If NumAut ≥ 2
    For i = 1,..., NumAut
        If IsThisBetter(i) // this function is described
          below
            Return False
2) Return True
END

IsThisBetter(x)
For i = 1... NumEdges
    If  a[Edges[i][1]][Edges[i][2]] <
        a[aut[Edges[i][1]]][aut[Edges[i][2]]]
        Return True
    If  a[Edges[i][1]][Edges[i][2]] >
        a[aut[Edges[i][1]]][aut[Edges[i][2]]]
        Return False
Return False
```

RESULTS AND DISCUSSIONS

This work was motivated by Miyake and Aids study of water clusters.[1,2] The largest clusters they have been able to treat are those on 8 vertices. The algorithm presented

here is able to go a step forward and up to now we have been able to enumerate topologically distinct clusters up to 12 vertices. Our algorithm is superior due to two facts:

1) it incorporates restrictive conditions on generation of unoriented graph into recursive function which results in much less number of generated objects,

2) it eliminates duplicates of digraphs using only automorphisms of the underlying graph instead of considering all permutations of vertices (hence, we consider only a few automorphisms of the graph instead of $n!$ permutations to be considered in the method of Miyake and Aida).

As an example of the advantages of our algorithm, Ref. 10 gives an example in which it is shown that our algorithm needs to consider only 15 697 041 matrices as compared to 68 719 476 736 matrices to be considered in the method of Miyake and Aida.

The results obtained by the algorithm described here are summarized in table given below:[10]

TABLE I. Number of graphs and digraphs describing water clusters with up to 12 water molecules

Vertex[(a)]	Graph[(b)]	Digraph[(c)]
2	1	1
3	2	5
4	6	22
5	21	161
6	78	1 406
7	353	14 241
8	1929	164 461
9	12207	2 115 335
10	89402	29 903 139
11	739335	460 066 726
12	6800637	7 644 586 673

[(a)]The number of the vertices.
[(b)]The number of the graphs generated.
[(c)]The number of the digraphs generated.

Acknowledgment. – Partial support of the Ministry of Science, Education and Sports of the Republic Croatia (Grants No. 098-0982929-2940 and No. 177-0000000-0884) is gratefully acknowledged.

REFERENCES

1. T. Miyake and M. Aida, *Chem. Phys. Lett.* **363** (2002) 106–110.
2. T. Miyake and M. Aida, *Internet Electronic Journal of Molecular Design* **2** (2003) 24–32.
3. D. J. Wales and M. P. Hodges, *Chem. Phys. Lett.* **286** (1998) 65–72.
4. I.-L. Kuo, J. V. Coe, S. J. Singer, Y. B. Band, and L. Ojamäe, *J. Chem. Phys.* **114** (2001) 2527–2540.
5. B. Bollobás, *Graph Theory*, Springer Verlag, Berlin, 1979.
6. A. Gibbons, *Algorithmic Graph Theory*, Cambridge University Press, Cambridge, 1985.
7. A. Graovac, I. Gutman, and N. Trinajstić *Topological Approach to the Chemistry of Conjugated Molecules,* Springer Verlag, Berlin, 1977.
8. N. Trinajstić, *Chemical Graph Theory*, CRC Press, Boca Raton, 1983; 2nd revised Ed., 1992.
9. I. Gutman and O. E. Polansky, *Mathematical Concepts in Organic Chemistry,* Springer Verlag, Berlin, 1986.
10. D. Vukičević, T. Grubeša, and A. Graovac, *Chem. Phys. Lett.* **416** (2005) 212–214.

SAŽETAK

Algoritam za prebrojavanje posebne klase usmjerenih grafova: primijena na klastere molekula vode

Damir Vukičević i Ante Graovac

Miyake i Aida su nedavno predložile model zasnovan na usmjerenim grafovima i razvile algoritam za prebrojavanje svih mogućih, topološki različitih, vodikovim vezama povezanih, grozdova molekula vode. Ovdje je prikazan novi algoritam temeljen na rekurzivnim funkcijama. Brojčani rezultati pokazuju da je ovaj znatno brži od Miyake-Aida algoritma.