J Math Chem (2009) 46:1112–1121
DOI 10.1007/s10910-008-9496-y

ORIGINAL PAPER

Generating water clusters and other directed graphs

Gunnar Brinkmann

Received: 16 June 2008 / Accepted: 17 October 2008 / Published online: 13 November 2008

Abstract In this paper we will describe an efficient method to generate directed graphs with bounded in- and out-degree. Though developed for the special case of water clusters, which can be modelled as directed graphs with indegree and outdegree at most 2 that have no two directed edges with the same endpoints, the algorithm is also applicable and implemented in the more general context.

Keywords Water · Cluster · Directed graph · Isomorphism · Degree

1 Introduction

Water molecules can bond to each other by hydrogen bonding. This bond is between a proton-donor and a proton acceptor, so it is equipped with a sense of direction. Describing this bond by a directed edge from the donor to the acceptor (see Fig. 1) translates a cluster of n water molecules to a directed graph with n vertices. Every molecule can give and accept two protons, but no molecule can have two bonds with the same other molecule (no matter whether it would donate two protons or donate one and accept one). We can characterize the directed graphs that model water clusters as follows:

Definition 1 A water cluster is a directed graph with maximum outdegree at most 2 and maximum indegree at most 2 which has no two directed edges with the same sets of endpoints.

G. Brinkmann (✉)
Department of Applied Mathematics & Computer Science, Ghent University, Krijgslaan 281-S9, 9000 Ghent, Belgium
e-mail: Gunnar.Brinkmann@UGent.be

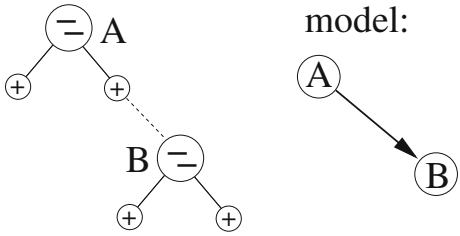

Fig. 1 A hydrogen bond encoded as a directed edge

In this article we will concentrate on the algorithmic aspects of this problem. We will not describe the physico-chemical background of waterclusters which has been discussed by other authors in previous papers. The reader is referred to e.g. [1] for an introduction and other aspects of the graph theory of water clusters.

The first algorithm to generate water clusters was by Miyake and Aida [2] and was used to enumerate all clusters on up to 8 molecules. Later this approach was improved by D. Vukičević et al. who could enumerate all clusters with up to 12 molecules [3,4]. The approach described in this article is several orders of magnitude faster than the faster one of these approaches. It takes 19 min on a single processor 2.6 GHz Pentium PC while the program by Vukičević et al. took 366 h on a cluster with 16 Pentium 2.4 GHz processors. The load factor is not given, but the 366 h given by the authors are the time between the start of the first part of the computation and the end of the last part—it is not the sum of the running times on the various processors.

Already at the time when the first generation programs for water clusters were published, a much faster and more general algorithm and program *directg* developed by B.D. McKay existed (see http://cs.anu.edu.au/~bdm/nauty/). *Directg* computes all non-isomorphic directed graphs for a given set of underlying undirected graphs. These underlying graphs can be generated by the program *makeg* by the same author. Already the most simple way to adapt this program to the needs of generating water clusters—filtering the output of *directg* to suppress graphs with too high in- or out-degree and without applying any early bounding criteria—gives a much faster program than those by Miyake and Aida, resp. Vukičević et al. (20 h for 12 vertices on a single processor 2.6 GHz Pentium PC). Including the degree restriction already in the generation process would further improve the performance of *directg*.

The algorithm and program described here will be even faster than *directg* also for cases without degree restrictions, so using branch and bound techniques and look aheads in *directg* to deal with the degree restrictions of water clusters without other modifications to the algorithm would not lead to a program as fast as the one described here.

2 The algorithm

All approaches to generate water clusters use the same basic strategy: generate all underlying graphs and direct the edges in all possible ways that give non-isomorphic graphs that fulfill the definition. The algorithm described here is about how to assign these directions as efficiently as possible.

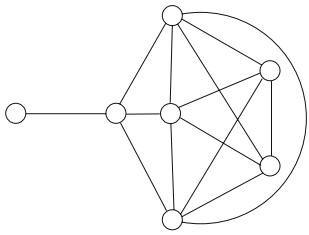

Fig. 2 An undirected graph that fulfills the characterization in Lemma 1 (a) for indegree at most 2 and outdegree at most 3 but does not allow any valid assignment of directions. This can be seen because the subgraph obtained by removing the degree 1 vertex does not fulfill the requirements of the characterization

Definition 2 We call a directed graph with maximal indegree at most x and maximal outdegree at most y that has no parallel directed edges (but possibly oppositely directed edges with the same endpoints) an (x, y)-digraph. An (x, y)-digraph with no two edges with the same endpoints is called a simple (x, y)-digraph.

In case of oppositely directed edges with the same endpoints we assume the underlying undirected graph to be the graph where these two directed edges correspond to a single undirected edge.

Lemma 1 *(a) The underlying graph G of an (x, y)-digraph with n vertices is a (simple) graph with n vertices, maximum degree $\Delta(G) \leq x + y$ and at most $\min\{x, y\} * n$ edges.*

(b) An undirected graph G with $\Delta(G) \leq 2 * \min\{x, y\}$ is the underlying graph of a simple (x, y)-digraph.

(c) In case $x = y$ each undirected graph with maximum degree at most $x + y = 2x$ is the underlying graph of a simple (x, y)-digraph.

Proof Part (a) is obvious and part (c) a direct consequence of (b), so we just have to prove part (b).
Let $z = \min\{x, y\}$. A graph G with maximum degree at most $2z$ is a subgraph of a $2z$-regular graph $\bar{G}$. But $\bar{G}$ has an Euler-cycle and directing the edges forward along this cycle gives a valid assignment of edges of G. □

Part (c) gives a justification of the approach using the undirected graphs described by Lemma 1 as the basis for assigning directions especially in case of $x = y$ and therefore also for water clusters. If a large fraction of the generated undirected graphs would not lead to any (x, y)-digraph, this approach would be less efficient.

But though in case $x \neq y$ there are undirected graphs that fulfill the requirements of Lemma 1 (a) but do not allow a valid assignment of directions (see Fig. 2) also in these cases the ratio of undirected graphs that do not allow a valid assignment of directions was very small—usually around 2% to 3% in the cases observed.

In fact the situation in Fig. 2 with a subgraph with too many edges is the only possibility that an underlying graph cannot be directed, as stated explicitly in the following lemma.

Lemma 2 *If for an undirected graph G every subgraph $G' = (V', E')$ has the property that $\Delta(G') \leq x + y$ and $|E'| \leq \min\{x, y\} * |V'|$, then the edges can be directed in a way to form an (x, y)-digraph.*

Proof In Lemma 1 this is already proven for $x = y$ so assume $x > y > 0$. The case $x < y$ can either be done analogously or can be obtained from $x < y$ by just reversing directions.

We prove it by induction in the number d of edges that have already been assigned a direction. It is obvious that for the first edge the direction can be chosen arbitrarily without violating the degree conditions, so assume that d edges are already directed and an edge $e = \{a, b\}$ is given that is to be directed.

In the first step we want to make sure that one of the vertices has an outdegree smaller than y. If this is already the case—w.l.o.g. for vertex a—we are done (with this step).

Otherwise both vertices have an indegree smaller than x because the degrees are at most $x + y$ and one adjacent edge is still undirected. Let V_a, V_b be the sets of vertices that are reachable from a, resp. b from a directed path. These sets can of course intersect or even be identical. If V_a and V_b both induce subgraphs of already directed edges with precisely $|V_a| * y$, resp $|V_b| * y$ edges, the set $V_a \cup V_b$ would have $|V_a| + |V_b| - |V_a \cap V_b|$ vertices and $|V_a| * y + |V_b| * y - |E(V_a \cap V_b)|$ edges with $|E(V_a \cap V_b)|$ the number of already directed edges with both endpoints in $V_a \cap V_b$. So by assumption $|E(V_a \cap V_b)| \leq |V_a \cap V_b| * y$ and therefore the number of edges is at least $|V_a| * y + |V_b| * y - |V_a \cap V_b| * y$. But then the subgraph induced by $V_a \cup V_b$ and including directed and undirected edges (so also e) violates the conditions in the lemma.

So assume that for V_a we have that the induced subgraph contains less than $|V_a| * y$ edges. Then there is a vertex v that is reachable by a directed path from a and has an outdegree less than y. Now we can reverse directions along this path, leaving indegree and outdegree unchanged at all interior vertices, increasing the outdegree of v and decreasing the outdegree of a.

So in any case the edge $\{a, b\}$ can now be directed $a \to b$ unless the indegree of b is x. But then we can apply the same argument with antidirected paths (paths that would be directed if all directions would be reversed) starting at b. Since $x > y$ we can now not only conclude that one of the subgraphs induced by the correspondingly defined $|V_a|$ and $|V_b|$ has less than $|V_a| * x$, resp $|V_b| * x$ edges, but both—so especially the one induced by V_b. Note that the outdegree of a is not increased during this operation, so that afterwards we can direct the edge $a \to b$ completing our proof. □

An obvious, but important observation is the following lemma:

Lemma 3

- *An isomorphism between two directed graphs induces an isomorphism of the underlying undirected graphs.*
- *If we start with an underlying graph and assign directions to the edges in two different ways (that is: there is at least one edge where the direction was assigned differently), any isomorphism of the resulting directed graphs induces a nontrivial automorphism of the graph.*

This lemma was also used by earlier approaches. It implies that we do not need any isomorphism rejection in case of underlying graphs with a trivial symmetry.

Two different ways to assign directions will always result in two non-isomorphic graphs. This is a special case of an isomorphism rejection technique called *homomorphism principle* (see [5]). A crucial step in this algorithm is to reach this ideal situation by splitting the 2-step construction (first generating underlying graphs and then directing them) into several steps. This technique is also responsible for the high generation rate in [6].

Because we start with an undirected graph G and assign directions to edges, we can interpret this as working with a labelling $l()$ of the edges. An edge $\{x, y\}$ can be labelled 0 (not yet decided), (x, y) (directed from x to y), (y, x) (directed from y to x) and $\{y, x\}$ (assigned two oppositely directed edges). In case of water clusters the last label $\{y, x\}$ may not be used. We denote such edge-labelled graphs as pairs (G, l) and will refer to the edges labelled 0 as *undirected*. The set of edges with nonzero label is called the domain of the labelling and we write dom(l).

Definition 3 A mapping $f : G \to G'$ between two edge-labelled graphs is called an isomorphism of the graphs if it is an isomorphism of the underlying unlabelled graphs and for every edge $\{x, y\}$ the relation between the label l of $\{x, y\}$ and l' of $\{f(x), f(y)\}$ is
$l = 0 \Leftrightarrow l' = 0$
$l = (x, y) \Leftrightarrow l' = (f(x), f(y))$
$l = (y, x) \Leftrightarrow l' = (f(y), f(x))$
$l = \{y, x\} \Leftrightarrow l' = \{f(y), f(x)\}$

If $G = (V, E)$ is a graph and l, l' are edge labelings so that for every edge $e \in dom(l)$ we have $l(e) = l'(e)$, then we call l' an extension of l.

Definition 4 An edge-labelled graph (G, l) is called *closed* if for all extensions l_1, l_2 of l any isomorphism between (G, l_1) and (G, l_2) induces an automorphism of (G, l).

The edge-labelled graph (G, l) with $l(e) = 0\ \forall e \in E$ is an example of a closed edge-labelled graph. In case of closed graphs with trivial automorphism group, we can proceed like in the special case of underlying undirected graphs with a trivial automorphism group: we can direct the not yet directed edges in every way compatible with the degree restrictions without having to test any isomorphisms—all directed graphs will be non-isomorphic. We split the labelling process into several steps as follows: For an undirected graph $G = (V, E)$ we construct a series of labelings $l_1, \ldots, l_k$ so that $dom(l_1) = \emptyset$, $dom(l_k) = E$, for $1 \leq i \leq k$, (G, l_i) is closed and for $1 \leq i < k$ the labelling l_{i+1} is a nontrivial extension of l_i.

Lemma 4 *Let (G, l) be a closed edge-labelled graph with $G = (V, E)$ and $D = dom(l)$. Furthermore let D' be an orbit of edges under the automorphism group of (G, l) and $l(e) = 0$ for all edges in D'.*

If l' is an extension of l so that $dom(l') = D \cup D'$, then (G, l') is also a closed edge-labelled graph.

Proof Assume extensions l_1, l_2 of l' and an isomorphism $f : (G, l_1) \to (G, l_2)$ are given.

Because (G, l) is closed, $f()$ induces an automorphism of (G, l). So $f()$ maps edges in D' onto other edges in D'. Since $D' \subseteq \mathrm{dom}(l_1), \mathrm{dom}(l_2)$ this means that $f()$ induces an automorphism of (G, l'). So (G, l') is closed. □

Now our strategy is clear:

a.) Generate all underlying undirected graphs $G = (V, E)$ with the properties given in Lemma 1.

b.) For each $G = (V, E)$ start with $l(e) = 0 \quad \forall e \in E$ and repeat the following loop until $\mathrm{dom}(l) = E$:

- b1.) Compute the automorphism group Γ of (G, l).
- b2.) If Γ is trivial, leave the loop and continue extending l without any isomorphism rejection.
- b3.) Else compute the orbits of undirected edges under Γ and choose a smallest orbit D'.
- b4.) If $D = \mathrm{dom}(l)$ then compute all extensions l' of l with $\mathrm{dom}(l') = D \cup D'$ that give non-isomorphic edge-labelled graphs.
  - If $\mathrm{dom}(l') = E$ output this graph.
  - else repeat the loop for $l = l'$.

For step a.) we use the program *makeg* (see [7]) and for step b1.) we used the program *nauty* (see [8]).

The only step for which it is not immediately clear how it can be done is b4.).

In case of an underlying graph with an automorphism group that acts transitively on the edges, this step contains the **whole** labelling process. The efficiency of the approach is based on the fact that cases where step b2.) cannot be applied or can only be applied when almost all edges are already labelled are very rare. For most classes of graphs the ratio of graphs with trivial group tends to 1 and already for small vertex numbers the ratio of graphs with few symmetries is very large. Of course the ratios vary depending on the class of graphs we are generating. For the class we are most interested in—water clusters—the relative amount of directed graphs that come from closed graphs with trivial symmetry and no edges directed, resp. at most 30% of the edges directed is depicted in Fig. 3. Note that also in the approach described here, most of the time is spent on dealing with graphs with symmetries: although more than 83% of the water clusters on 12 vertices come from underlying graphs with a trivial symmetry, only 20% of the time is spent on assigning labels to these and 80% of the time to label the remaining 17% that come from underlying graphs with symmetries.

To assign directions to edges of a chosen orbit, McKay's *canonical construction path method* (see [7]) is used. It is a standard technique, so it will just be sketched here. Assume we have a closed partially directed graph $(G = (V, E), l)$ with $D = \mathrm{dom}(l)$ and an orbit D' of undirected edges is given. We want to compute all isomorphism classes of graphs (G, l') so that l' is an extension of l and $\mathrm{dom}(l') \subseteq D \cup D'$. Let us call this class $X(G, l, D')$.

We will generate the class by starting with (G, l) and assigning directions to one edge at a time. The canonical construction path method now requires

a.) to assign a unique ancestor with one directed edge less to every graph in $X(G, l, D')$ except for the base graph (G, l).

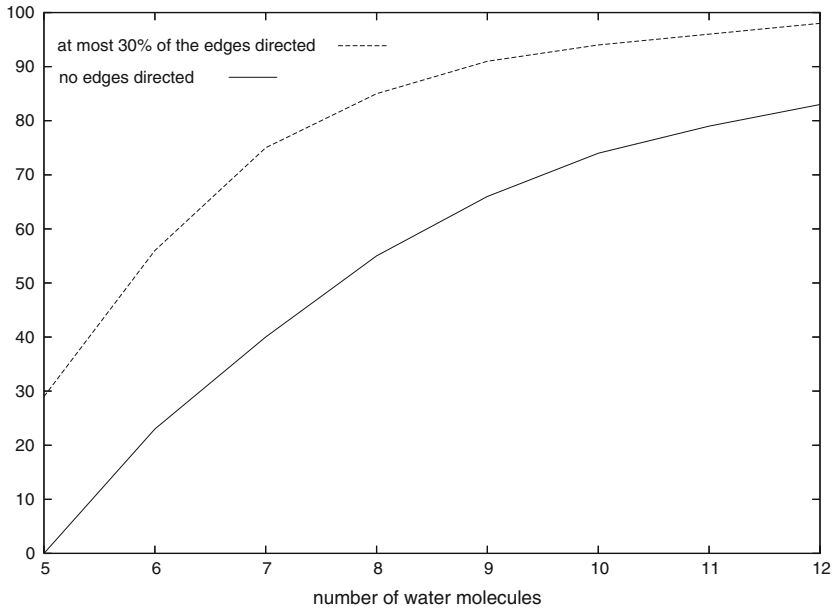

Fig. 3 The percentage of water clusters coming from a closed graph with 12 vertices and no, resp. at most 30% of the edges directed

b.) to make sure that from no graph in $X(G, l, D')$ two isomorphic ancestors are generated.

Part a.) is implemented by canonically choosing an orbit of directed edges in D'. The ancestor is the graph obtained by removing the direction of one of these edges. Note that the isomorphism class of the graph obtained this way does not depend on which edge in the orbit is chosen. The graph is accepted as *being properly generated* if and only if the last directed edge belongs to the canonical orbit. In order to do this efficiently, first some easily computable vertex invariants are used to assign numbers to the directed edges in a way that is invariant under automorphisms. These are based on the in- and out-degree of the vertices and orbit numbers of the vertices assigned by *nauty()* for the graph (G, l). In case this cheap criterion already gives a unique edge in D' with a minimal invariant, we have found our canonical edge and therefore our ancestor. Otherwise we have to use *nauty()* to compute a canonical labelling and the automorphism group in order to decide on an orbit. As a means to speed up the computation, orbit numbers determined by *nauty()* for (G, l) are used as vertex colours for the graph (G, l') that is tested for canonicity.

Part b.) is implemented by computing for every graph (G, l') that is to be extended the automorphism group and the orbits of undirected edges in D' under this group. Then only one edge in every orbit is chosen to be directed and produce a descendant with one edge more. It is easy to see that directing another edge in the same orbit instead of the chosen one would give an isomorphic result (so we don't miss graphs). On the other hand it can be seen that if two descendants of (G, l') would be isomorphic

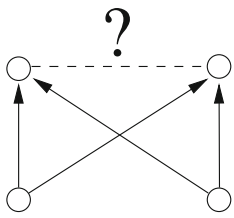

Fig. 4 A partial assignment of directions for water clusters that cannot be completed

and both accepted, the isomorphism can be chosen in a way that the edges that were the last ones to be directed are mapped onto each other. This induces an automorphism of (G, l') mapping the two undirected edges onto each other—so two edges from the same orbit would have been assigned directions.

This gives the following lemma:

Lemma 5 *For a given closed graph* (G, l) *and orbit* D' *of undirected edges, the described method generates exactly one representative for every isomorphism class of graphs in* $X(G, l, D')$.

According to Lemma 1, in case of water clusters every undirected graph generated can be assigned directions to form a water cluster. But for degree restrictions with different in- and out-degrees this is not true and even for water clusters not every partial assignment of directions can be completed (see Fig. 4).

Computing expensive criteria to determine whether a partial assignment of directions can be completed, would slow down the computations. On the other hand it is obvious that the chance to be able to extend a partial assignment of directions to still undirected edges around vertices with degree d is higher for small d—in case of $d \leq \min\{i, o\}$ (with i the bound for the indegree and o the bound for the outdegree) it is even sure that the assignment can be extended to all edges around the vertex.

In case of symmetries, the order in which edges to direct are chosen is given by the isomorphism rejection routines, but in case of trivial symmetry of the closed graph, we can freely choose the order in which we direct the remaining undirected edges. We place the edges on a stack (so the edges that are first placed are the last ones to be directed) in the following way:

- Repeat until the graph is empty:
  - Choose a vertex v of minimum degree.
  - Place all edges incident with v on the stack.
  - Remove v and all incident edges from the graph.

For water clusters with 12 vertices, this order of assigning directions to edges in graphs with trivial symmetry was 2.8 times faster than a random order or a lexicographic order of the edges. This factor is increasing with the number of vertices.

3 Testing

The output of the program *water* based on the described algorithm was tested in various ways. First of course the previously known results for water clusters on up to 12 vertices were compared.

In addition, *directg* was used to assign directions in every possible way to all graphs on up to 7 vertices and all bipartite graphs on 8 or 9 vertices. The output was analysed to determine the numbers of directed graphs based on these classes for all combinations of maximal indegree, maximal outdegree and presence of edges directed in both directions. Then for every combination *water* was started and the number of graphs generated for this restricted class was compared. Note that the whole class is also one of the cases tested. In total 560 cases were tested and the results were in perfect agreement.

It should be noted that the test against *directg* is not completely independent—both programs use *nauty* and *makeg*. But since *nauty* and *makeg* are widely used and therefore extensively tested, this overlap can be accepted.

4 Results

The most interesting numbers are surely those of water clusters. These are given in Table 1. Assuming an average load factor of 10 of the cluster on which the

Table 1 The numbers of water clusters and underlying graphs and the running times of *water* on an Intel Xeon processor with GNU/linux and 2.66 GHz

Molecules	Undirected graphs	Water clusters	Time (s)	
1	1	1		
2	1	1		
3	2	5		
4	6	22		
5	21	161		
6	78	1,406		
7	353	14,241		Miyake, Aida
8	1,929	164,461	0.07	
9	12,207	2,115,335	1.1	
10	89,402	29,903,139	5.7	
11	739,335	460,066,726	23	Vukičević et al.
12	6,800,637	7,644,586,673	644	
13	68,531,618	136,336,779,596	5.680	
14	748,592,936	2,596,190,669,230	61.800	
15	8,788,983,173	52,552,267,768,902	1.201.300	
16	110,201,690,911	1,126,421,027,176,730		

The jobs were run on an 8 processor machine and split into 6 parts for 14, into 60 parts for 15 and into 600 parts for 16 vertices. The times given are the sums of the times of the parts

Table 2 The numbers of directed graphs without degree restrictions for some classes of underlying graphs and the running times of *water* and *directg*

Underlying class of undirected graphs	Number of directed graphs	Water (seconds)	Directg (seconds)	Factor
$\|V\| = 6$	1.540.944	0.36	10.8	30
$\|V\| = 7$	882.033.440	32.8	12,023	366
$\|V\| = 8$, bipartite	5.578.632	2.3	54.3	23
$\|V\| = 9$, $\Delta \leq 3$	65.917.619	3.5	45.7	12.8
$\|V\| = 10$, bipartite and $\Delta \leq 3$	53.611.591	5.1	65.3	12.7
$\|V\| = 11$, bipartite connected, $\Delta \leq 4$	174.531.268.392	1,985	144,091	72.6

The times are given for a 2.6 GHz Pentium 4 processor with GNU/linux. In cases like these where no degree bounds are given, the program can sometimes determine the numbers of graphs without actually constructing them, but for the timings all graphs were explicitly formed in memory

computations for [3] were performed, *water* is approximately 10.000 times faster for 12 vertices.

Because *directg* is designed for more general purposes and not equipped with an efficient test of the degree restrictions already during the construction, comparing *water* with *directg* for cases with degree restrictions would give a wrong impression. So Table 2 only gives a comparison for classes without degree restrictions.

The source code of the program *water* can be obtained for academic purposes free of charge from the author.

References

1. J.L. Kuo, J.V. Coe, S.J. Singer, Y. Band, L. Ojamäe. On the use of graph invariants for efficiently generating hydrogen bond topologies and predicting physical properties of water clusters and ice. J. Chem. Phys. **114**, 2527 (2001)
2. T. Miyake, M. Aida. Enumeration of topology-distinct structures of hydrogen bonded water clusters. Chem. Phys. Lett. **363**, 106 (2002)
3. D. Vukičević, T. Grubeša. A. Graovac. An efficient method to enumerate topologically distinct clusters of hydrogen-bonding in water molecules. Chem. Phys. Lett. **416**, 212 (2005)
4. D. Vukičević, A. Graovac. An algorithm to enumerate a special class of digraphs: application to water clusters. Croatica Chemica. Acta **2**(81), 347 (2008)
5. G. Brinkmann. Isomorphism rejection in structure generation programs., in ed. by P. Hansen, P.W. Fowler, M. Zheng. *Discrete Mathematical Chemistry*. vol. 51 of DIMACS Series on Discrete Mathematics and Theoretical Computer Science (American Mathematical Society, 2000), pp. 25–38
6. G. Brinkmann, A.A. Dobrynin, A. Krause, Fast generation of polycyclic chains with arbitrary ring sizes. MATCH Commun. Math. Comput. Chem. **41**, 137 (2000)
7. B.D. McKay, Isomorph-free exhaustive generation. J. Algorithm. **26**, 306 (1998)
8. B.D. McKay, Practical graph isomorphism. Congressus Numerantium **30**, 45 (1981)