International Journal of Chemical Modeling
Volume 5, Number 2-3, pp. 67–89
ISSN 1941-3955

THE HISTORY OF THE GENERATION OF CUBIC GRAPHS

Gunnar Brinkmann, Jan Goedgebeur and Nico Van Cleemput
Ghent University, Department Applied Mathematics,
Computer Science and Statistics, Ghent, Belgium

Keywords: cubic graphs, generation

AMS Subject Classification: 01-02, 05-03, 05C30

1. Introduction

A cubic graph is a simple graph in which each vertex is adjacent to three other vertices. Cubic graphs have since long received much attention from both mathematicians and chemists. In chemistry, cubic graphs can be used to model carbon networks, since carbon often binds with three C atoms. This makes lists of cubic graphs interesting as lists of possible molecules that can be studied for their chemical properties. Examples of this are the famous fullerenes [1] and cyclopolyenes [2]. For several important open conjectures in mathematics, it can be proven that if they are wrong, the smallest counterexample is a cubic graph. This makes lists of cubic graphs interesting as a possible source for counterexamples.

For these reasons constructing lists of cubic graphs has been a longstanding goal for mathematicians and chemists. The first complete lists date back to the end of the 19th century, when de Vries published a list of all cubic (connected) graphs on up to 10 vertices [3, 4].

The next step, and the first one using a computer, was taken by Balaban in 1966/67 when he generated all cubic graphs on up to 12 vertices [2].

Even after Balaban's work – which was obviously not known to many mathematicians – still manual approaches were performed [5, 6] that were only able to confirm de Vries' results, but did not go as far as Balaban's computer enumeration.

Starting with Balaban's work, faster and faster algorithms were proposed to generate complete lists of cubic graphs so that – of course also due to the improvement of computers – now [7] hundreds of thousands of non-isomorphic cubic graphs can be generated per second. As the lists are intended for evaluating the graphs and testing conjectures on them, a further improvement does not seem to be useful: one can generate much larger lists than one can reasonably store and practically each test that is performed on the lists takes longer than the generation.

The sequence of papers of algorithms also show some other development: while in the beginning the emphasis was on the numbers and the algorithms were hardly or not at all

Table 1. The number of connected cubic graphs on a given number of vertices as determined by Robinson and Wormald [8]

Vertices	Graphs
4	1
6	2
8	5
10	19
12	85
14	509
16	4 060
18	41 301
20	510 489
22	7 319 447
24	117 940 535
26	2 094 480 864
28	40 497 138 011
30	845 480 228 069
32	18 941 522 184 590
34	453 090 162 062 723
36	11 523 392 072 541 432
38	310 467 244 165 539 782
40	8 832 736 318 937 756 165

described, later the algorithms were described in more and more detail and also the running times were given in a more exact way.

2. The Manual Approach

Neither for the first computational approach nor for the first manual approach to generate cubic graphs, cubic graphs as mathematical structures were the goal.

In the 19th century de Vries was interested in geometrical structures known as *configurations*, but these special configurations are in fact equivalent to cubic graphs. De Vries investigated plane configurations in which each point is the intersection of two lines and each line contains three points. In Figure 1 the relation between plane configurations and cubic graphs is shown: we can interpret the lines as the vertices of the graph, and the points as the edges of the graph.

For his construction de Vries applied a set of three operations which were more than 100 years later also the base of the fastest computer algorithm. He recursively applied these operations to the set of plane configurations he already had generated, starting with the plane configuration shown in Figure 1 on the left. Though he was working with geometric structures, he interpreted them in a purely combinatorial way – that is: he described which

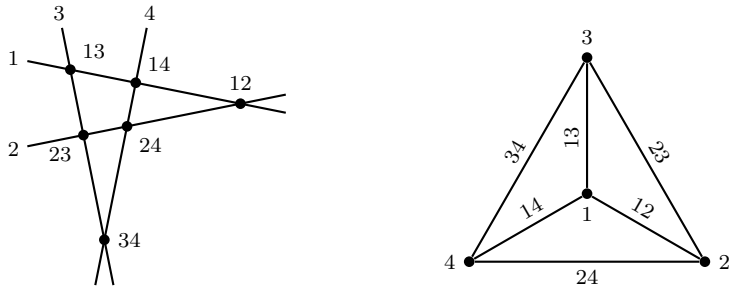

Figure 1. The relation between a plane configuration (left) and a cubic graph (right).

edges must intersect which other edges without discussing the question whether this is geometrically possible. The operations that were used by de Vries are the following:

1. Given two points cd and ef that do not lie on the same line. Then two new lines – a and b – are added that cross in a new point ab.

   On the line c, respectively d, the point cd is replaced by the point ac, respectively ad, and on the line e, respectively f, the point ef is replaced by the point be, respectively bf. This operation is shown in Figure 2 (a).

2. Given a line a. The points ag, respectively ah, are replaced by the points ac and cg, respectively ad and dh. Finally the point cd is added to the plane configuration, so that each line contains three points. This operation is shown in Figure 2 (b).

3. Given a point bi. Four new lines (a, c, d and g) with five new points (ac, ad, cd, cg and dg) are added to the plane configuration, while at the same time, the point bi is replaced by ab on b and by gi on i. This operation is shown in Figure 2 (c).

In Figure 3 the graph equivalents of the de Vries operations are given.

Using his three operations, de Vries manually succeeded in constructing all plane configurations with up to 10 lines. He found 1 configuration with 4 lines (shown on the left in Figure 1), 2 configurations with 6 lines, 5 configurations with 8 lines and 18 configurations with 10 lines. These are the correct numbers, except for the last number, as can be seen in Table 1: there are 19 graphs with 10 vertices. In the French version of his article, de Vries corrects this mistake and gives the nineteenth configuration. The missing configuration is shown in Figure 4. Using de Vries' operations, this configuration can only be obtained by applying the first operation to the points 34 and 78 in the disconnected plane configuration in Figure 5. He originally only considered the connected plane configurations, which explains why he initially missed this nineteenth configuration.

De Vries' results were manually confirmed in a note by Bussemaker and Seidel in 1968 [5] and by an article by Imrich in 1971 [6]. We could not obtain the note by Bussemaker and Seidel, so we cannot report on their methods. In his article, Imrich also mentions computer work by Baron done at the Technische Hochschule Wien in 1966, where all cubic graphs on 10 vertices were generated. Based on this research, Imrich posed the question whether the graphs can also be *easily* determined by hand and said that based on the structure of the

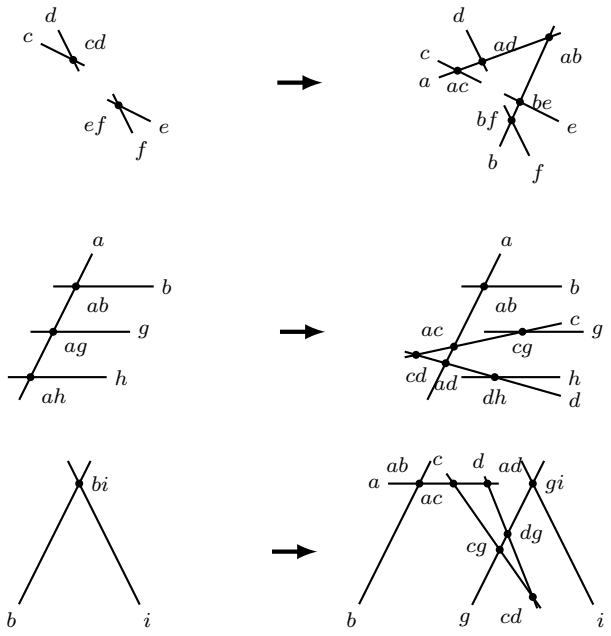

Figure 2. The three operations used by de Vries.

19 graphs found by Baron a decomposition into classes seems useful. Imrich splitted the generation into cases and generated the graphs without triangles, with only one triangle, two disjoint triangles, two disjoint quadrangles with a diagonal and *other cases*. Inside the cases he used some arguments to restrict the number of connections that had to be evaluated, but essentially went through all possible connections. He also did not write how he detected isomorphisms and only wrote, e.g., about the 6 graphs without triangles: "*We omit to show that these 6 graphs are pairwise non-isomorphic.*"

Cubic graphs with 12 vertices formed the next step. In this case, however, there are already 85 graphs. This number of graphs is too large to perform all the calculations manually. Especially avoiding isomorphic copies becomes much more difficult. Therefore the next steps in the generation of cubic graphs could only be made with the help of computers.

3. Computer Era

1966/67: Balaban: 10/12 Vertices

The first documented computer-assisted generation of cubic graphs was by Balaban in 1966/67. In [6] Imrich reported about a computer search by Baron in 1966, but we could not find any documentation of this. Balaban generated all cubic graphs on up to 10 vertices [2], and later also added the cubic graphs on 12 vertices. Balaban was not studying graphs, but he was looking for all the valence-isomers of cyclopolyenes $C_{2p}H_{2p}$. He noted that the case where there are no double bonds, and each C atom is bonded with 3 other C atoms, is equivalent to the problem of the generation of cubic graphs. He also noted that this problem

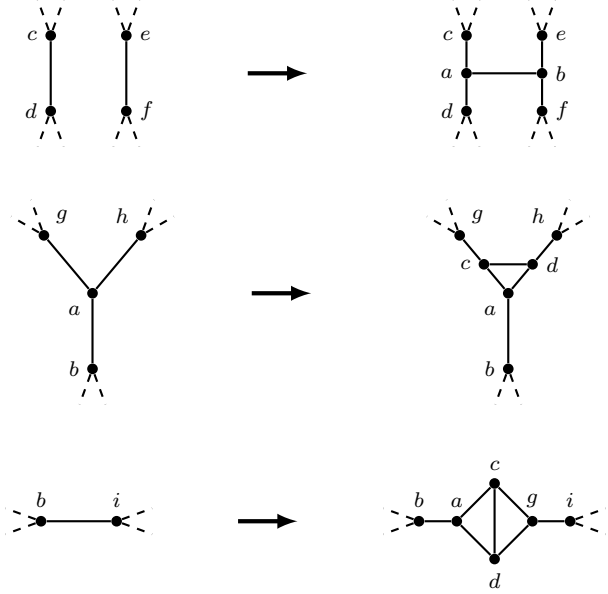

Figure 3. The three operations used by de Vries translated to a graph context.

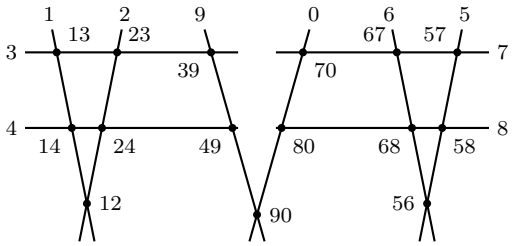

Figure 4. The plane configuration which was missing in the original paper by de Vries.

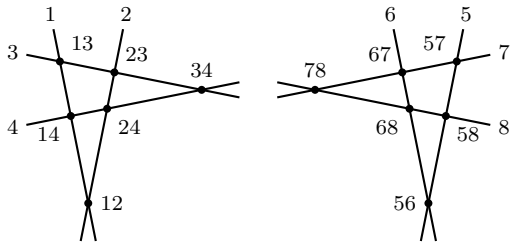

Figure 5. The only disconnected 3-regular plane configuration with 8 lines.

is not yet solved in mathematics. To generate all cubic graphs, Balaban used several brute-force techniques. He started with a cycle containing the $2p$ vertices and then made the remaining connections, but also connected vertices and chains of vertices to each other. The details of these methods are not given. The detection of isomorphic copies was assisted by looking at the number of cycles of length 3, 4 and 5 of each graph, but the exact method of isomorphism rejection is also not described.

1974: Petrenjuk and Petrenjuk: 12 Vertices

In 1974 Petrenjuk and Petrenjuk have generated all cubic graphs on 12 vertices again [9]. Unfortunately their paper (in Russian) is hard to get, so we can not report on their methods. We only know of their work as it is cited in [10].

1976: Bussemaker, Čobeljić, Cvetković, Seidel: 14 Vertices

In 1976, Bussemaker, Čobeljić, Cvetković and Seidel released a catalog of all cubic graphs on up to 14 vertices [11]. They listed for each graph the structure, the characteristic polynomial, the eigenvalues, the number of cycles for each possible length, the diameter, the connectivity, the planarity, the size of the automorphism group and a picture of the graph. They also mentioned how some other properties can easily be deduced from the given properties, such as the hamiltonicity, the girth and the chromatic number. The graphs were generated using a program developed by Bussemaker. For the details they refer to a separate technical report which unfortunately never appeared. They did mention how the program was tested: the number of graphs on 12 vertices was independently calculated by Čobeljić using a heuristic hand-computer search. Furthermore they also calculated the number of labelled cubic graphs directly and compared the results to the unlabelled counts by using the orders of the automorphism groups. They also point out several incorrect numbers in older results and identify the errors that were made. In [12] they also compared their own numbers to those of Faradžev.

The labellings of the graphs published suggest that they also used the canonical form later used by Brinkmann in [13, 14] and most likely also Faradžev in [10], but this is not described in the article.

1976: Faradžev: 18 Vertices

In 1976 Faradžev published the numbers of cubic graphs on up to 18 vertices [10]. Together with a paper by Read [15], his paper is considered the foundation of the method of orderly generation. The paper does not describe the algorithm for the construction of cubic graphs in detail, but uses the results on cubic graphs and some other graph classes as an example. The main focus lies on describing a very abstract meta-algorithm that became later known as orderly generation.

Besides connected cubic graphs, Faradžev also gives lists of connected regular graphs of degree 4, 5, and 6, connected bipartite regular graphs of degree 3,4, and 5 and numbers of 3-vertex-connected graphs. About the programs the article only says that *"the above described method was applied to the solution of the constructive enumeration problem for some classes of graphs"*.

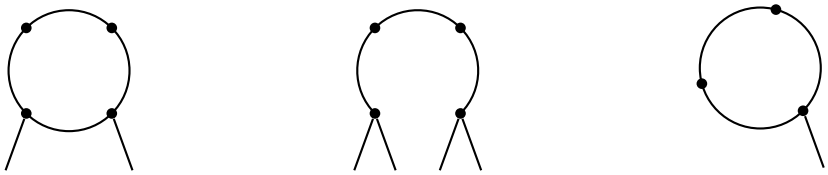

Figure 6. From left to right an example of an ear of type 0, respectively of type 1 and of type 2. All these ears have length 3.

It is not clear in how far the generator for cubic graphs coincides with the one for, e.g., 4-regular graphs and in how far the generators were optimized for the specific graph classes. It is also not said how exactly the abstract conditions were implemented. It seems likely that he uses a *canonical form* later also used for the *minibaum*-program, but that is not clear from the article.

In the end Faradžev also says for which values he could compare his results with previously known results – this is important information to judge how well tested the programs were.

1986: McKay and Royle: 20 Vertices

Ten years later, in 1986, McKay and Royle raised the bar once again by generating all cubic graphs on 20 vertices [16]. Though this article focusses on the generation of cubic graphs and also describes the details of the algorithm, its importance is – just like in the case of Faradžev's algorithm – also rooted in the fact that it introduced a new meta-algorithm for isomorphism rejection. This method – known as the "canonical construction path method" was ten years later published in a more abstract way as a general method for isomorph-free structure generation [17]. For each graph G that arises during the generation process, an (up to isomorphism) unique reduction operation giving a parent $p(G)$ is defined and G is accepted if and only if it is generated by the extension operation that is the inverse of the unique reduction operation. Furthermore it is guaranteed that for each parent graph no two extension operations that are equivalent under the automorphism group are applied.

McKay and Royle started from the 2-regular graphs and the disconnected graphs with 3- or 2-regular components such that at least one component is 2-regular. In a graph with vertex degrees 2 and 3, an ear of length k is a sequence $v_0, v_1, \ldots, v_k$ such that v_{i-1} and v_i are adjacent (for $1 \leq i \leq k$), $v_1, \ldots, v_{k-1}$ have degree 2 and v_0 and v_k have degree 3, and the v_i are distinct, except possibly $v_0 = v_k$. If v_0 and v_k are adjacent the ear is of type 0, if $v_0 = v_k$ the ear is of type 2, and in all other cases the ear is of type 1. Examples of these ears are shown in Figure 6. The internal vertices of an ear are $v_1, \ldots, v_{k-1}$ if $v_0 \neq v_k$, and all vertices otherwise.

To construct all cubic graphs they added ears to the start graphs and intermediate graphs as long as these graphs had vertices of degree 2. For each graph G that is to be extended, only extensions are applied that are not equivalent under the automorphism group of G – that is: where the endpoints of the ears to be attached are not mapped onto each other by an automorphism. In case the graph contained exactly three vertices of degree 2, they also

added a single vertex and connected it to the three vertices of degree 2.

They defined a vertex invariant they call *quality* as the number of vertices at distance 2 plus 100 times the number of vertices at distance 3. Furthermore they used a canonical labelling [18, 19]– that is a way to assign labels to vertices that is unique up to automorphisms of the graph.

The parent of a graph was defined as follows:

- the start graphs do not have a parent;
- for a cubic graph, the parent is the graph obtained by deleting the vertex with the lowest quality and if this is not unique, among them one with the lowest canonical label;
- for all other intermediate graphs, the parent is the graph obtained by selecting from the set of ears of lowest type, the subset of those of greatest length and then removing the internal vertices of the ear from that subset which contains the vertex with the lowest canonical label.

This method already contains all ingredients that make later algorithms using the canonical construction path method so efficient.

1992: Brinkmann: 24 Vertices

The program *minibaum* by Brinkmann [13, 14] uses an orderly algorithm like described by Faradžev. The algorithm is tailored for cubic graphs and uses some datastructure exploiting similarities of graphs that are generated to speed up isomorphism rejection. In 1992 the program was only used for all cubic graphs on up to 24 vertices, but in the meantime – due to faster computers and large clusters – it was used for up to 32 vertices.

For each graph $G = (V,E)$ with $V = \{1,\ldots,|V|\}$, the adjacency matrix $(a_{i,j})_{1\leq i,j,\leq |V|}$ is given by $a_{i,j} = 1$ if $\{i,j\} \in E$ and $a_{i,j} = 0$ otherwise. One can concatenate the rows of this 0-1-matrix to form one long binary representation of a natural number. The result is called the order number of the graph. The permutation of vertex labels that leads to the largest order number is called the *canonical labelling* of the graph and a graph with maximum order number is called a maximum representative.

For each isomorphism class of cubic graphs, *minibaum* constructs only the unique maximum representative. These graphs are constructed by inserting the edges in increasing lexicographical order – so the first edges inserted are always $\{1,2\},\{1,3\},\{1,4\}$ as they are present in each maximum cubic graph and are lexicographically smallest. If a maximum graph is constructed by inserting the edges in lexicographical order, then all intermediate graphs are also maximum (see [14]) which leads to the following algorithm for generating all cubic graphs with vertex set $V = \{1,\ldots,n\}$, which is an orderly algorithm in the sense of Read and Faradžev:

- Start with the graph $(V,\{\{1,2\},\{1,3\},\{1,4\}\})$.
- Recursively add edges in lexicographical order. Each edge inserted is adjacent to the smallest vertex that still has a degree smaller than 3.

- After an edge is inserted, check whether the graph is a maximum representative by trying to find a labelling that leads to a larger representation. If a larger representation is found: reject the graph, otherwise proceed.

- If a graph is found to be maximum and all vertices have degree 3: output the graph.

It can easily be proven that the first edge inserted that closes a cycle determines the girth of a graph. If later an edge is inserted that lies on a smaller cycle, the graph is not maximum. This observation avoids the construction of some graphs that are afterwards found to be not maximum.

The main technique that speeds up the program (and is also responsible for the name of the program) can be understood as follows:

For a given graph G in the construction process that is not yet 3-regular, all graphs constructed from G contain G as a subgraph. During the test whether G is maximum, it may be detected that one gets smaller results when trying to assign label 1 to vertex x already before assigning labels to a vertex with degree less than 3. If this is the case, the same will happen for all descendants of G, so one can in fact store the information that it is useless to assign label 1 to vertex x for all descendants and avoid these computations.

This idea can be carried further: one can store all partial assignments of labels to vertices of degree 3 that do not yet lead to smaller representations and start for the descendants from the partial assignments instead of from scratch. This information has to be updated in each step and therefore an efficient datastructure has to be used to store the information. The program *minibaum* uses a tree (German: Baum) of these partial representations.

As updating the datastructure is more expensive than just testing, this tree is only built up to a certain depth in the recursion where the cost is smaller than the benefit. This depth had to be determined by tests.

When run on an Intel(R) Core(TM)2 Quad CPU Q8200 with 2.0 Ghz, *minibaum* generates about 81.500 non-isomorphic cubic graphs on 26 vertices per second.

An advantage of the simple edge-by-edge insertion strategy is that each graph constructed is a subgraph of all its descendants. So all graph properties that are inherited by the descendants can be – more or less – efficiently included in the generation process. This way, e.g., graphs without small cycles, or without cycles of a given length (e.g., without odd lengths – that is: bipartite graphs) can be generated. Unfortunately for very restricted classes of cubic graphs (e.g., graphs with large girth), isomorphism rejection is not the problem any more, but most partial graphs cannot be completed, what then leads to very small generation rates.

1998: Meringer: 24 Vertices

In 1998 Meringer [20] used the same basic orderly algorithm as Brinkmann in [13, 14] for his program *genreg* but optimised the test whether a graph is maximum so that it also works efficiently for regular graphs with larger degree, while the method in [14] gets inefficient for larger degrees.

Some of the key new ideas used come from [21] where graphs with a given degree sequence are generated, but unlike Meringer's program, Grund's program was not optimized for regular graphs.

Meringer only tested the regular graphs that were to be output for being maximum and used the concept of *semicanonicity*, which is a necessary, but not sufficient criterion for a graph to be maximum for the intermediate graphs. This criterion is less selective, but much faster to test. It only tests whether certain subsets of the whole permutation group can produce a graph with a larger representation. These subsets are defined recursively. First all permutations are chosen that fix vertex 1, among those only those that fix the edges adjacent to 1, among those only those that fix vertex 2, etc.

The result of this faster but less restrictive criterion can be that an intermediate graph that is not maximum could produce a large number of descendants that will all be tested and rejected for being not maximum. This problem is solved by *learning* from elements that are not maximum: if a regular graph is detected to be not maximum, the permutation giving a smaller representation often gives a smaller representation already when applied to a subgraph. If e denotes the lexicographically largest edge that occurs in the subgraph or the image of the subgraph, the smaller labeling will be possible as long as all edges up to e are present. So Meringer's algorithm backtracks until the edge e is removed, as graphs containing all edges up to e will not be maximum anyway.

These two key ideas are combined with the use of a Sims chain of nested subgroups of the whole permutation group allowing to compute a set of generators instead of going through all automorphisms when trying to find smaller labellings.

The result is a program that – though not designed especially for cubic graphs, but for regular graphs of given degree – showed a good performance for cubic graphs and is the fastest program for degrees 4 or larger where it could extend some lists of graphs with or without restricted girth. Like *minibaum* also *genreg* offers the option to only generate graphs with restricted girth or bipartite graphs.

When run on an Intel(R) Core(TM)2 Quad CPU Q8200 with 2.0 Ghz, *genreg* generates about 50.100 non-isomorphic cubic graphs on 26 vertices per second.

2000: Sanjmyatav and McKay

In 2000 Sanjmyatav and McKay [22] developed new generation algorithms for various classes of cubic graphs. The construction operations of their algorithm to generate all cubic graphs are based on the operations from de Vries. Their two basic operations are shown in Figure 3(a) and Figure 3(b) and are called the *edge* and *triangle operation*, respectively. The inverse of a construction operation is called a reduction. Cubic graphs which cannot be reduced to a smaller cubic graph by a triangle or edge reduction were called *irreducible graphs*. By definition each cubic graph can be generated from an irreducible graph by recursively applying edge and/or triangle operations.

They showed that all irreducible graphs can be obtained by recursively applying operation (c) from Figure 3 (where the vertices b and i may be the same) to cubic multigraphs with loops. As only a tiny fraction of the cubic graphs are irreducible (see Table 2), they used a simple straightforward method to generate irreducible graphs in advance. They produced a list of irreducible graphs on up to 26 vertices which were then used as a starting point for the generation of all cubic graphs on up to 26 vertices.

For generating the remaining cubic graphs from these irreducible graphs, they used the canonical construction path method to avoid the generation of isomorphic copies. This

means that they computed orbits of possible extensions (orbits of pairs of edges to apply operation (a) and orbits of vertices to apply operation (b)) and used a canonicity criterion for the unique reduction of a graph that first uses some easily computable invariants and falls back to a canonical labeling [18] in case this did not give a unique reduction to decide whether the last operation is canonical in the sense of the canonical construction path method.

Table 2. Number of irreducible graphs vs. number of cubic graphs

$\|V(G)\|$	# irreducible graphs	# cubic graphs
4	1	1
6	0	2
8	1	5
10	1	19
12	1	85
14	3	509
16	2	4 060
18	5	41 301
20	4	510 489
22	9	7 319 447
24	11	117 940 535
26	16	2 094 480 864
28	32	40 497 138 011
30	37	845 480 228 069
32	73	18 941 522 184 590

They also designed algorithms to generate more restricted classes of cubic graphs such as bipartite cubic graphs or cubic graphs with girth at least 4 or 5. For most of these classes they did this in two ways. In one approach they added a filter and look-aheads for the restricted graph class to the generation algorithm for all cubic graphs. In the other approach they developed a specialized construction algorithm which constructs cubic graphs from the special class in such a way that all intermediate graphs are also part of the special class. In most of these cases the filter approach was the more efficient one.

Unfortunately the programs were never released and we could not compare them on modern machines. The times reported in the thesis are that the generator applied to generate all cubic graphs is approximately 2.7 times faster than *minibaum* for 20 vertices and about 1.7 times faster for 24 vertices.

This ratio may differ on modern machines as different algorithms may react differently to faster or slower memory. They give, e.g., the same time (52 seconds) for the generation of cubic graphs with 20 vertices by *minibaum* and *genreg*, while on an Intel(R) Core(TM)2 Quad CPU Q8200 with 2.0 Ghz the times differ by more than 20%: 7 seconds for *minibaum* and 9 seconds for *genreg*.

2011: Brinkmann, Goedgebeur, McKay: 32 Vertices

In 2011 Brinkmann, Goedgebeur and McKay [7] developed a new program called *snarkhunter* which allowed to generate all cubic graphs up to 32 vertices. The construction operations are based on the operations from de Vries and Sanjmyatav/McKay and the edge and triangle operations are again applied to generate cubic graphs. Nevertheless there were also several points where the algorithms differ. The new algorithm used some special optimisations to define the canonical reverse operations and generated irreducible graphs (also called prime graphs) in a different way. It was shown that every prime graph can be obtained by recursively applying the operations from Figure 7 to K_4 in such a way that all intermediate graphs are also prime. Both for the generation of prime graphs and for the generation of the remaining cubic graphs the canonical construction path method was used to make sure no isomorphic copies are output.

The main new idea leading to a speedup was to handle graphs with isolated triangles in another way. A *reducible triangle* is a triangle which can be reduced by a triangle reduction. As most cubic graphs contain at least one reducible triangle (see Table 3), special attention was paid to generating cubic graphs with reducible triangles. As a rule used for the canonical construction path method, graphs with reducible triangles must be constructed by expanding vertices to triangles. The new algorithm *bundled* these triangle operations. For the reduction of graphs with reducible triangles, the idea was to reduce all reducible triangles at the same time. In the construction, the bundled triangle operation blows up all vertices in a set $S \subseteq V(G)$ of a graph G to triangles at the same time in such a way that all reducible triangles in the graph obtained by applying this bundled triangle operation are created in this last operation. This means that S must contain at least one vertex of every reducible triangle in G. Orbits of vertex sets to be extended must be computed to avoid isomorphic copies – but this is done on much fewer and much smaller graphs. A small problem is formed by a square attached to two triangles at opposite edges of the square, as these triangles can be reduced separately, but not in parallel. Graphs with such a configuration – which does not occur often – come from graphs with a square that has a diagonal. The ratio of these graphs is very small and this case was handled by separate rules.

In order to make sure that only one graph is output for each isomorphism class of cubic graphs, a canonicity check is performed and non-canonical graphs are rejected. This can be very expensive in general. However since the graph obtained by applying the bundled triangle reduction to a cubic graph with reducible triangles is uniquely determined, no canonicity check is required for cubic graphs with reducible triangles and no non-canonical graphs are generated by the bundled triangle operation. This significantly sped up the algorithm.

The bundled triangle operation also helps to compute the automorphism group of the extended graph, which is needed in case the resulting graph has not yet reached the desired number of vertices. In case of a trivial group of the graph to be extended, it can even be concluded that the extended graph also has a trivial group.

When run on an Intel(R) Core(TM)2 Quad CPU Q8200 with 2.0 Ghz, *snarkhunter* generates about 410.300 non-isomorphic cubic graphs on 26 vertices per second. Based on the ratios with *genreg* and *minibaum* given in [22] this is considerably faster than the algorithm described there.

The algorithm can also be restricted to generate cubic graphs with girth at least 4 or 5

Table 3. Counts of all 2 094 480 864 connected cubic graphs with 26 vertices according to the number of reducible triangles they have

# reducible triangles	# graphs
0	497 010 000
1	774 885 044
2	540 977 972
3	218 274 256
4	54 459 966
5	8 183 373
6	666 137
7	23 837
8	279

efficiently by using look-aheads. Similar look-aheads could also be used to generate cubic graphs with larger lower bounds on the girth, but these look-aheads would be a lot less efficient. For generating cubic graphs with large girth, the fastest method is that of McKay, Myrvold and Nadon [23].

The main reason for the development of *snarkhunter* was the possibility to restrict the generation to snarks – that is: to cyclically 4-connected cubic graphs with chromatic index 4 – in a way that is more efficient than filtering all graphs. Based on the lists of snarks generated, 8 published conjectures could be refuted [24].

4. Counting the Graphs

All methods described in the previous paragraphs were constructive, i.e., all the graphs were explicitly formed and could be output and used for tests. In mathematics *"enumeration"* is also used for determining only the number of structures, but not the structures themselves. Enumerating cubic graphs in this sense, one can go much further than would be possible by constructing all the graphs. This was done by Robinson and Wormald in 1983 [8]. They used a counting technique which allowed them to calculate the number of cubic graphs on up to 40 vertices. Even at that time their program needed about 150 hours to calculate these numbers.

5. Restrictions and Generalisations

5.1. Restricted Classes of Cubic Graphs

As already mentioned before, several restricted classes of cubic graphs are also of mathematical and chemical interest. Unlike, e.g., bipartite cubic graphs or cubic graphs with

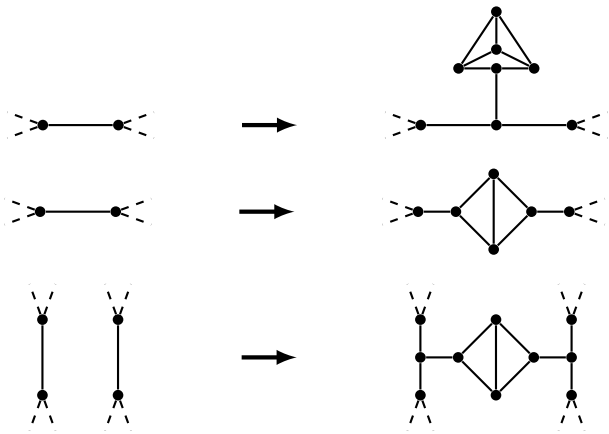

Figure 7. The three operations used by Brinkmann, Goedgebeur and McKay to construct prime graphs.

girth 4 or 5, some of these classes can not efficiently be generated by modifying the generators for all graphs. We will not go into details of generation algorithms for these restricted classes, but just mention some of them.

As Balaban points out in [25], planar cubic graphs are, e.g., more likely to correspond to actual molecules and already in 1972/1976 he gave lists of small cubic planar graphs [26, 25] by filtering the cubic graphs he had generated. The fastest available algorithm at the moment also goes back to a construction method from the 19th century [27]. Together with an efficient algorithm for isomorphism rejection it is implemented in the program *plantri* [28]. On an Intel(R) Core(TM)2 Quad CPU Q8200 with 2.0 Ghz, *plantri* needs about 1.85 seconds to generate all cubic polyhedra (3-connected cubic planar graphs) on 26 vertices – which are about 1.300.000 polyhedra on 26 vertices per second.

Things get more complicated when the class of cubic polyhedra is further restricted. After the discovery of the Fullerenes (that is: carbon cages with the structure of cubic polyhedra with all faces of size 5 or 6) in 1985 [1], generators for combinatorial fullerenes were needed and after a series of incomplete algorithms [29] in 1997 the first complete and efficient algorithm – *fullgen* – was published [30] and later generalized for arbitrary face sizes in [31]. The fastest fullerene generator nowadays is *buckygen* described in [32]. On an Intel(R) Core(TM)2 Quad CPU Q8200 with 2.0 Ghz it generates about 38.000 non-isomorphic fullerenes on 100 vertices per second.

In mathematics there are several inductive definitions known for several classes of cubic graphs, i.e., 2-connected cubic graphs, planar cubic graphs, 3-connected planar cubic graphs, see [33] or the references in [28]. These inductive definitions give rise to generation programs for specialized classes (included in the program described in [28]), but we won't describe the details here.

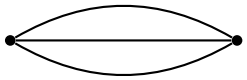

Figure 8. The only connected cubic multigraph with an edge with multiplicity 3.

5.2. Generalised Cubic Graphs

Besides restricted classes of cubic graphs, also generalisations were considered. Up to this point, we have only been discussing results for connected simple graphs. A canonical generalisation is to also allow multigraphs, graphs with loops and graphs with semi-edges, or any combination of these.

From a chemical viewpoint, multigraphs are a very natural generalisation to consider, since atoms can form double bonds. And it was also Balaban to do the first computer enumerations: In 1966 he enumerated all cubic multigraphs on up to 10 vertices and in 1967 he also added the cubic multigraphs on 12 vertices. He was interested in these graphs since they serve as models for the cyclopolyenes $C_{2p}H_{2p}$ that contain double bonds.

A cubic multigraph can have edges with multiplicity 1, 2 or 3. There is only one connected cubic multigraph with an edge with multiplicity 3. This is the theta graph (shown in Figure 8). All other cubic multigraphs can only contain edges with multiplicity 1 and 2. Balaban generated these graphs based on the number of vertices and the number of edges with multiplicity 2.

For the case with $2p$ vertices, p edges with multiplicity 1 and p edges with multiplicity 2, he noted that there is only one multigraph. This is the cycle on $2p$ vertices where the edges alternatingly have multiplicity 1 and 2.

For the case with $2p$ vertices, $p+2$ edges with multiplicity 1 and $p-1$ edges with multiplicity 2, he divided the graphs into three categories and, based on the number of vertices, gave closed formulas for the number of graphs in each of these categories. The three categories are:

A. The cycles with one chord (see Figure 9 for an example); The number of graphs in this category is $\left\lfloor \frac{p-1}{2} \right\rfloor$.

B. The cycles with an extra path between two vertices (see Figure 10 for an example); The number of graphs in this category is $\left\lfloor \frac{p^2-2p+4}{12} \right\rfloor$

C. Two cycles connected by a path (see Figure 11 for an example). The number of graphs in this category is $\left\lfloor \frac{p-1}{2} \right\rfloor \left\lceil \frac{p-1}{2} \right\rceil$

For the remaining cases, Balaban used the same techniques as for cubic simple graphs, i.e., all possible ways to connect the vertices were tried.

In 1970 Balaban revisited the generation of cubic multigraphs as part of the generation of general cubic graphs [34, 35, 36]. A general graph is a multigraph that can also have loops. To generate all general cubic graphs with n vertices, he applied two operations to the general cubic graphs on $n-2$ vertices. These operations are:

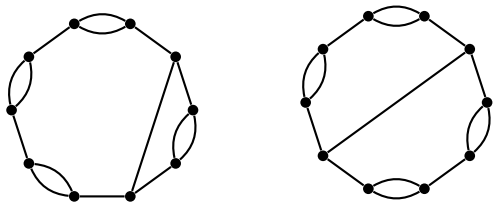

Figure 9. The two multigraphs on 10 vertices with 7 edges of multiplicity 1 and 4 edges of multiplicity 2, that belong to category A.

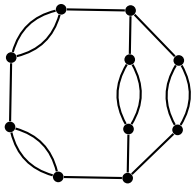

Figure 10. The only multigraphs on 10 vertices with 7 edges of multiplicity 1 and 4 edges of multiplicity 2, that belong to category B.

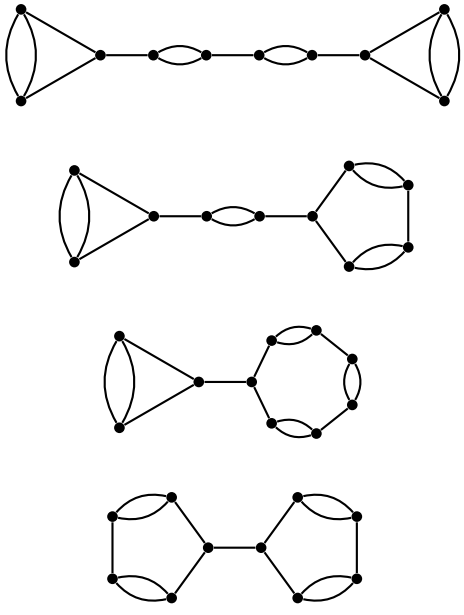

Figure 11. The four multigraphs on 10 vertices with 7 edges of multiplicity 1 and 4 edges of multiplicity 2, that belong to category C.

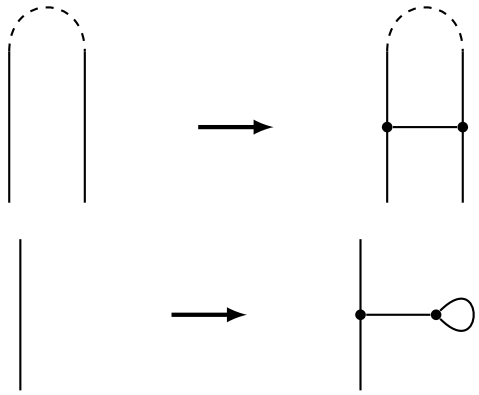

Figure 12. The two operations used by Balaban to generate general cubic graphs.

1. Add two new vertices to any edge(s) or loop(s) and connect them by a new edge (see top part of Figure 12).

2. Add a new vertex to any edge or loop and connect it by a new edge to another new vertex to which also a loop is added (see bottom part of Figure 12).

This technique was used to generate all general cubic graphs on up to 12 vertices, starting from the two general cubic graphs on two vertices: the theta graph (see Figure 8) and the K_2 with an extra loop incident to each vertex. The cubic multigraphs were then obtained by taking the general cubic graphs with 0 loops. This technique generates many graphs several times, so the produced list of graphs was afterwards filtered for duplicate graphs.

In 2012 Brinkmann, Pisanski and Van Cleemput developed a generation algorithm for several generalised classes of cubic graphs [37]. It can generate graphs that can have loops, semi-edges and multi-edges. A semi-edge is an edge that is incident to only one vertex. The algorithm can be efficiently restricted to graphs allowing only a subset of these non-simple edge types.

The problem of generating these generalised graphs is first translated to the generation of *cubic pregraph primitives*. A cubic pregraph primitive is a multigraph with vertices of degree 1 and 3. The vertices of degree 1 ultimately give rise to loops and/or semi-edges, depending on the class that is generated. To generate all cubic pregraph primitives on n vertices, the canonical construction path method is applied again. It uses 4 extension operations and as base graphs K_2 , the theta graph and all cubic graphs on up to n vertices. The 4 operations are depicted in Figure 13.

In order to generate the cubic pregraphs that have loops or semi-edges from these cubic pregraph primitives, a loop is added to each vertex of degree 1, or a vertex of degree 1 together with the adjacent edge is replaced by a semi-edge to the neighbour. In order to generate the cubic pregraphs that have both – loops and semi-edges – from these cubic pregraph primitives, the homomorphism principle [38, 39] is used to avoid isomorphic copies. For small vertex numbers the numbers obtained coincide with the numbers Balaban already calculated in 1970. The number of structures for several classes and vertex numbers are

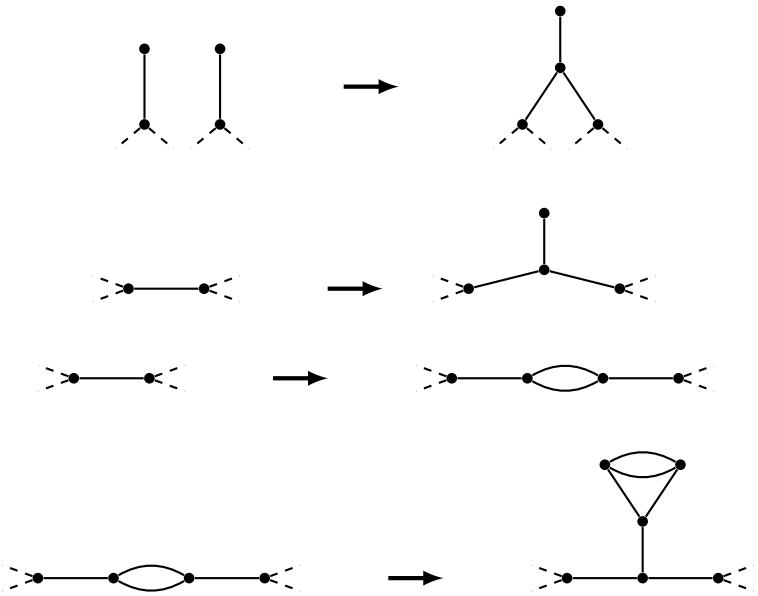

Figure 13. The four operation used by Brinkmann, Pisanski and Van Cleemput to generate cubic pregraph primitives.

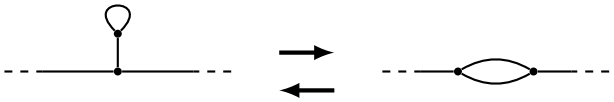

Figure 14. An occurrence of a loop can be transformed into an edge of multiplicity 2 and vice versa.

shown in Table 4.

When run on an Intel(R) Core(TM)2 Quad CPU Q8200 with 2.0 Ghz, the program *pregraphs* based on this approach generates about 78.000 non-isomorphic cubic multigraphs with loops and semi-edges on 16 vertices per second.

One observation that can be made when looking at the number of cubic pregraphs is that the numbers in column L, respectively LS, coincide with the numbers in column M, respectively SM (except for $n = 1$). This correspondence was already noted and explained by Balaban. The reason is that each occurrence of a loop can be transformed into an edge of multiplicity 2 and vice versa (see Figure 14). This is always possible except for the balloon graph (see Figure 15), which explains the difference for $n = 1$.

Table 4. The number of structures in each class of cubic pregraphs for a given number of vertices *n*. C stands for simple graphs, L for graphs that have loops, S for graphs that have semi-edges, M for multigraphs, LS for graphs that have loops and semi-edges, LM for multigraphs that have loops (i.e., general graphs), SM for multigraphs that have semi-edges and LSM for all pregraphs

n	C	L	S	M	LS	LM	SM	LSM
1	0	0	1	0	2	0	1	2
2	0	1	1	1	3	2	3	5
3	0	0	2	0	4	0	4	7
4	1	2	6	2	12	5	12	22
5	0	0	10	0	22	0	22	43
6	2	6	29	6	68	17	68	141
7	0	0	64	0	166	0	166	373
8	5	20	194	20	534	71	534	1 270
9	0	0	531	0	1 589	0	1 589	4 053
10	19	91	1 733	91	5 464	388	5 464	14 671
11	0	0	5 524	0	18 579	0	18 579	52 826
12	85	509	19 430	509	68 320	2 592	68 320	203 289
13	0	0	69 322	0	255 424	0	255 424	795 581
14	509	3 608	262 044	3 608	1 000 852	21 096	1 000 852	3 241 367
15	0	0	1 016 740	0	4 018 156	0	4 018 156	13 504 130
16	4 060	31 856	4 101 318	31 856	16 671 976	204 638	16 671 976	57 904 671
17	0	0	16 996 157	0	70 890 940	0	70 890 940	253 856 990
18	41 301	340 416	72 556 640	340 416	309 439 942	2 317 172	309 439 942	1 139 231 977
19	0	0	317 558 689	0	1 381 815 168	0	1 381 815 168	5 219 113 084
20	510 489	4 269 971	1 424 644 848	4 269 971	6 310 880 471	30 024 276	6 310 880 471	24 401 837 085
21	0	0	6 536 588 420	0	29 428 287 639	0	29 428 287 639	116 278 408 069
22	7 319 447	61 133 757	30 647 561 117	61 133 757	140 012 980 007	437 469 859	140 012 980 007	564 380 686 932
23	0	0	146 647 344 812	0		0		
24	117 940 535	978 098 997		978 098 997		7 067 109 598		

Figure 15. The balloon graph is the only graph in which the loop cannot be transformed into an edge of multiplicity 2.

Conclusion

The large number of different and increasingly fast approaches to generate cubic graphs gives a hint about the importance of this class. In the long series of computational approaches there are always two special ones:

the first approach and the last approach. While the last approach is only waiting for the next algorithm that is even faster, the first approach will always remain the first – and for cubic simple graphs as well as cubic multigraphs, it will always be the name of Balaban standing for these approaches.

Not many examples of scientists can be found that not only made important contributions at young age, but also – with more than 80 years and more than 50 years after their Ph.D. – still contribute innovative scientific ideas. A.T. Balaban is one of the few. We want to express our gratitude for the honour to have met this outstanding scientist on several conferences.

References

[1] H.W. Kroto, J.R. Heath, S.C. O'Brien, R.F. Curl, and R.E. Smalley. C_{60}: Buckminsterfullerene. *Nature*, 318:162–163, 1985.

[2] Alexandru T. Balaban. Valence-isomerism of cyclopolyenes. *Revue Roumaine de chimie 11*, pages 1097–1116, 1966. No. 12 (1967) p.103 (erratum).

[3] J. de Vries. Over vlakke configuraties waarin elk punt met twee lijnen incident is. *Verslagen en Mededeelingen der Koninklijke Akademie voor Wetenschappen, Afdeeling Natuurkunde* (3) 6, pages 382–407, 1889.

[4] J. de Vries. Sur les configurations planes dont chaque point supporte deux droites. *Rendiconti Circolo Mat. Palermo 5*, pages 221–226, 1891.

[5] F.C. Bussemaker and J.J. Seidel. Cubical graphs of order $2n \leq 10$. *T.H. Eindhoven*, Note No.10, September 1968.

[6] W. Imrich. Zehnpunktige kubische Graphen. *Aequationes Math.* 6, pages 6–10, 1971.

[7] G. Brinkmann, J. Goedgebeur, and B.D. McKay. Generation of cubic graphs. *Discrete Mathematics and Theoretical Computer Science*, 13(2):69–80, 2011.

[8] R.W. Robinson and N.C. Wormald. Numbers of cubic graphs. *Journal of Graph Theory, Vol. 7*, pages 463–467, 1983.

[9] A.N. Petrenjuk and L.P. Petrenjuk. On constructive enumeration of 12 vertex cubic graphs (russian). *Combinatorial analysis, no.3 Moscow*, 1974.

[10] I.A. Faradžev. Constructive enumeration of combinatorial objects. *Colloques internationaux C.N.R.S. No260 - Problèmes Combinatoires et Théorie des Graphes, Orsay 1976*, pages 131–135, 1976.

[11] F.C. Bussemaker, S. Čobeljić, D.M. Cvetković, and J.J. Seidel. Computer investigation of cubic graphs. Technical Report T.H.Report 76-WSK-01, Department of Math. Technological University Eindhoven, The Netherlands, 1976.

[12] F.C. Bussemaker, S. Čobeljić, D.M. Cvetković, and J.J. Seidel. Cubic graphs on $\leq$14 vertices. *J. Combinatorial Theory Ser. B* 23, pages 234–235, 1977.

[13] G. Brinkmann. Generating cubic graphs faster than isomorphism checking. Technical report, SFB 343, Universität Bielefeld, 1992.

[14] G. Brinkmann. Fast generation of cubic graphs. *Journal of Graph Theory*, 23(2):139–149, 1996.

[15] Ronald C. Read. Every one a winner. *Annals of Discrete Mathematics* 2, pages 107–120, 1978.

[16] Brendan D. McKay and Gordon F. Royle. Constructing the cubic graphs on up to 20 vertices. *Ars Combinatoria 21 a*, pages 129–140, 1986.

[17] B. D. McKay. Isomorph-free exhaustive generation. *Journal of Algorithms*, 26:306–324, 1998.

[18] B. D. McKay. Practical graph isomorphism. In *10th. Manitoba Conference on Numerical Mathematics and Computing* (Winnipeg, 1980), volume 30 of *Congressus Numerantium*, pages 45–87, 1981.

[19] B. D. McKay. nauty User's Guide (version 1.5). Technical Report TR-CS-90-02, Australian National University, Department of Computer Science, 1990. http://cs.anu.edu.au/~bdm/nauty.

[20] M. Meringer. Fast generation of regular graphs and construction of cages. *Journal of Graph Theory*, 30(2):137–146, 1999.

[21] R. Grund. Konstruktion schlichter Graphen mit gegebener Gradpartition. *Bayreuther Mathematische Schriften*, 44:73–104, 1993.

[22] S. Sanjmyatav. Algorithms for generation of cubic graphs. Master's thesis, Australian National University, 2000. promotor: B.D. McKay.

[23] D. McKay B. W. Myrvold, and J. Nadon. Fast backtracking principles applied to find new cages. In *9th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 188–191, January 1998.

[24] G. Brinkmann, J. Goedgebeur, J. Hgglund, and K. Markstrm. Generation and properties of snarks. to appear in Journal of Combinatorial Theory, Series B.

[25] Alexandru T. Balaban. *Chemical Applications of Graph Theory*, chapter Enumeration of Cyclic Graphs, pages 63 – 105. Academic Press, 1976.

[26] Alexandru T. Balaban. ??? *Revue Roumaine de Chimie*, 17:865, 1972.

[27] V. Eberhard. *Zur Morphologie der Polyeder*. Teubner, 1891.

[28] G. Brinkmann and B.D. McKay. Fast generation of planar graphs. *MATCH Commun. Math. Comput. Chem.*, 42(4):909–924, 2007. see http://cs.anu.edu.au/∼bdm/index.html.

[29] P.W. Fowler and D.E. Manolopoulos. *An Atlas of Fullerenes*. Oxford University Press, 1995.

[30] G. Brinkmann and A.W.M. Dress. A constructive enumeration of fullerenes. *Journal of Algorithms*, 23:345–358, 1997.

[31] G. Brinkmann, O. Heidemeier, and T. Harmuth. The construction of cubic and quartic planar maps with prescribed face degrees. *Discr. Appl. Math.*, 128(2–3):541–554, 2003.

[32] G. Brinkmann, J. Goedgebeur, and B.D. McKay. The generation of fullerenes. *Journal of Chemical Information and Modeling*, 52(11):2910–2918, 2012.

[33] V. Batagelj. Inductive classes of cubic graphs. In *Colloquia Mathematica Societatis János Bolyai, 37. Finite and infinite sets*, pages 89–101, 1981. Eger.

[34] Alexandru T. Balaban. ??? *Revue Roumaine de chimie*, 15:463, 1970.

[35] A.T. Balaban. Chemical graphs. xxiii. general cubic graphs with one loop, and homovalenes or their derivatives. *Revue Roumaine de Chimie*, 19:1611–1619, 1974.

[36] A.T. Balaban. Erratum. chemical graphs. xxiii. general cubic graphs with one loop, and homovalenes or their derivatives. *Revue Roumaine de Chimie*, 23:311, 1978.

[37] Gunnar Brinkmann, Nico Van Cleemput, and Tomaž Pisanski. Generation of various classes of trivalent graphs. *Theoretical Computer Science*, 2012.

[38] R. Grund, A. Kerber, and R. Laue. MOLGEN – ein Computeralgebrasystem für die Konstruktion molekularer Graphen. *MATCH Commun. Math. Comput. Chem.*, 27:87–131, 1992.

[39] G. Brinkmann. Isomorphism rejection in structure generation programs. In P. Hansen, P.W. Fowler, and M. Zheng, editors, *Discrete Mathematical Chemistry*, volume 51 of *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, pages 25–38. American Mathematical Society, 2000.

MA