Discrete Mathematics and Theoretical Computer Science DMTCS vol. **13:2**, 2011, 69–80

Generation of Cubic graphs

Gunnar Brinkmann[1†] Jan Goedgebeur[1‡] Brendan D. McKay[2§]

[1] *Applied Mathematics & Computer Science, Ghent University, Belgium*
[2] *Research School of Computer Science, Australian National University, Canberra, Australia*

received 19th January 2011, *revised 28th June 2011*, *accepted 28th June 2011*.

We describe a new algorithm for the efficient generation of all non-isomorphic connected cubic graphs. Our implementation of this algorithm is more than 4 times faster than previous generators. The generation can also be efficiently restricted to cubic graphs with girth at least 4 or 5.

Keywords: graph, cubic graph, generation, canonical construction path

1 Introduction

Cubic or *3-regular* graphs are (simple) graphs where each vertex has degree 3. The class of cubic graphs is especially interesting for mathematical applications because for various important open problems in graph theory cubic graphs are the smallest or simplest possible potential counterexamples. In chemistry, cubic graphs serve as models for the Nobel Prize winning fullerenes [14] or, more generally, for some cyclopolyenes [2].

The generation of cubic graphs can be considered a benchmark problem in structure enumeration. The first complete lists of cubic connected graphs were given by de Vries at the end of the 19th century, who gave a list of all cubic (connected) graphs up to 10 vertices [9, 10]. The first computer approach was by Balaban, a theoretical chemist, in 1966/67 who generated all cubic graphs up to 12 vertices [2]. De Vries' lists were independently confirmed by hand by Bussemaker and Seidel in 1968 [8] and Imrich in 1971 [13]. From 1974 on, various algorithms for the generation of cubic graphs were published. Each algorithm was implemented in a computer program that could generate larger lists of cubic graphs, see [21, 11, 7, 18, 3]. In 1983 Robinson and Wormald [23] published a paper on the non-constructive enumeration of cubic graphs.

When the present research was begun, the fastest publicly available program for the generation of cubic graphs was *minibaum* [3]. When developed in 1992, *minibaum* could be used to generate complete lists of all cubic graphs up to 24 vertices and several restricted classes with more vertices, like bipartite graphs or graphs with higher girth. Later, when more and faster computers were available, minibaum was used to generate all cubic graphs up to 30 vertices in order to test them for Yutsis decompositions [1].

† Gunnar.Brinkmann@UGent.be
‡ Jan.Goedgebeur@UGent.be
§ bdm@cs.anu.edu.au

In 1999 Meringer [20] published a very efficient algorithm for the generation of regular graphs of given degree, but for the generation of all cubic graphs the program based on his algorithm is slower than *minibaum*. In 2000, Sanjmyatav [24] and her supervisor McKay developed a set of very fast specialized programs for various classes of cubic graphs. Unfortunately these programs were never released or published. In this article we will describe an algorithm based on ideas already used by de Vries and Sanjmyatav/McKay but also with some new ideas. Our implementation of this algorithm is much faster than any previous program.

2 The Generation Algorithm

Our basic construction operation to make a cubic graph G' with n vertices from a cubic graph G with $n-2$ vertices is the insertion of a new edge between new vertices inserted in two different edges of G. This operation was already used by de Vries and can be seen in Figure 1. The inverse operation involves removing an edge $\{x, y\}$ and its endpoints x and y, then adding an edge between the two vertices other than y that were previously adjacent to x, and an edge between the two vertices other than x that were previously adjacent to y. We call this an *edge reduction operation* in case the resulting graph is a connected cubic graph and then we call the edge $e = \{x, y\}$ *reducible*, otherwise *irreducible*. So e is irreducible if and only if it is a bridge, has an endpoint in a triangle that does not contain e, or has two endpoints in the same 4-gon that does not contain e.

A cubic connected graph without reducible edges is called a *prime graph*. By definition, each connected cubic graph can be constructed from a prime graph by recursive application of the edge insertion operation, so our first task will be to identify the prime graphs.

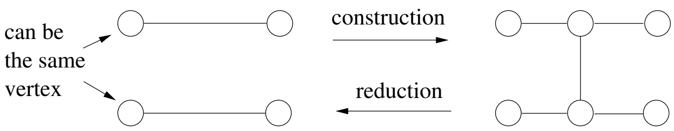

Figure 1: The basic edge insertion operation

We will refer to a subgraph of a graph isomorphic to $K_4 - e$ (with e an edge of K_4) as a K_4^-. The two vertices in a K_4^- that have degree 2 in this subgraph are called extremal vertices. K_4 with an edge subdivided by inserting a vertex of degree 2 is referred to as K_4^+.

Lemma 2.1 *A cubic graph is a prime graph if and only if each edge is a bridge or has an endpoint in a* K_4^-.

Proof: The direction $\Leftarrow$ is immediate, because neither bridges nor edges with endpoints in a K_4^- can be reduced.

So suppose that $e = \{x, y\}$ is an edge in G which is is not a bridge and has no endpoints in a K_4^-. If e has no endpoints in a triangle, then e is reducible. On the other hand, if x is contained in a triangle, but not in a K_4^-, each edge of the triangle is reducible. □

We will now use this characterization of prime graphs:

Lemma 2.2 *The class of prime graphs can be generated from* K_4 *by recursively applying the construction operations in Figure 2 in such a way that all intermediate graphs are also prime graphs.*

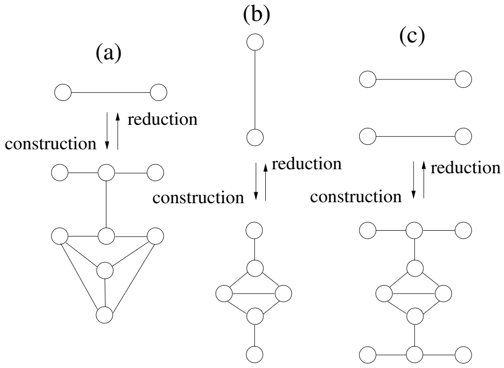

Figure 2: The construction operations for prime graphs.

Proof: We have to show that each prime graph G with more than 4 vertices can be reduced to a smaller prime graph by one of the reductions in Figure 2.

Assume first that G contains an edge e with both endpoints in different copies of a K_4^+ or a K_4^-. If one of the endpoints is in a K_4^-, operation (b) can be applied to this K_4^-. It is easy to see that the new edge has an endpoint in a K_4^- or is a bridge and that the other edges remain bridges or keep their endpoints in a K_4^-. In case both endpoints are in a K_4^+, the graph can only be the graph obtained by applying operation (a) to K_4.

Assume now that there are no edges with both endpoints in different copies of a K_4^+ or a K_4^-. Then each edge not contained in a K_4^- contains only one endpoint in a K_4^- or is a bridge. The existence of a K_4^+ implies that it can be reduced by reduction (a). The new edge will either be a bridge or have endpoints in two K_4^-, so the reduced graph is a prime graph.

If there are no subgraphs K_4^+, then there is an edge e with one endpoint in a K_4^- that lies on a cycle. Then reduction (c) can be applied to the K_4^- containing a vertex of e. The resulting graph will be connected because e was not a bridge and will be a prime graph because the new edges will be bridges resp. contain vertices of a K_4^- if the edges formerly incident with the vertices did. $\square$

Note that the class of prime graphs is not closed under these construction operations, so after applying an operation it must be tested whether the new graph is a prime graph or not.

We will now construct the class of all connected cubic graphs in two steps:

$$\boldsymbol{K_4}$$

$\Downarrow$ (operations (a),(b),(c))

Prime graphs

$\Downarrow$ (edge insertion)

All cubic graphs

As Table 1 shows, the number of prime graphs is completely negligible compared to the number of cubic graphs so it hardly matters what method is used to make them. Our implementation uses the canonical construction path method [16], which will be described in relation to the second step.

Though it is unimportant in practice, we can prove the asymptotic rarity of prime graphs. By induction using Lemma 2.2, each prime graph of order $2n$ for $n \geq 3$ has at least $n/3$ copies of K_4^-. Applying

$\|V(G)\|$	# prime 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

Table 1: Number of prime graphs vs. number of cubic graphs

reduction (b) of Figure 2 simultaneously to all copies of K_4^-, we obtain a cubic multigraph (parallel edges and loops allowed) on less than $2n/3$ vertices. This shows that the number of prime graphs of order $2n$ is at most equal to the number of connected cubic multigraphs of order less than $2n/3$. It was shown by Read [22] that the numbers of labeled cubic graphs or cubic multigraphs on $2k$ vertices is, in each case, asymptotically

$$\frac{\Theta(1)(6k)!}{288^k(3k)!}.$$

We can divide by $(2k)!$ to obtain the asymptotic counts of unlabeled graphs, since in both classes most graphs have trivial automorphism groups [19]. This shows that the fraction of cubic graphs which are prime is $n^{-\Omega(n)}$.

3 Efficient implementation of the edge insertion operation

The class of connected cubic graphs is closed under the edge insertion operation, so the only time consuming routines are those that make sure that only pairwise non-isomorphic graphs are generated. We will describe these in more detail. We use the canonical construction path method described in [16] but do not assume here that the reader is familiar with the method.

The theory of random cubic graphs, see [25] for example, says that the number of triangles is asymptotically a Poisson distribution with mean $\frac{4}{3}$, but the expected number of copies of K_4^- is $o(1)$. This implies that asymptotically about 74% of cubic graphs have triangles whose edges are reducible. This percentage is also approximately true for the small sizes we are dealing with, which justifies paying special attention to these *reducible triangles* as they allow some optimizations that do not work in general.

3.1 Generating graphs with reducible triangles

Note that the operation of edge insertion applied to two vertices having a common endpoint can be seen as that of replacing the common endpoint by a triangle, and the result depends only on what the common endpoint was and not which two edges were used. We call this *triangle insertion*. Similarly, the reverse operation of *triangle reduction* can be seen as that of replacing a reducible triangle by a vertex.

We give triangle reductions priority over other edge reductions in the sense that if a graph has reducible triangles we require it to be constructed by triangle insertion. This allows us to *bundle* triangle operations. The idea is to reduce each reducible triangle at the same time, but there is a twist in that the two triangles in the subgraph $ext(K_4^-)$ shown in Figure 3 cannot be reduced at once. However reduction of either of the two triangles in an $ext(K_4^-)$ produces the same smaller graph, so we define our bundled triangle reduction as simultaneously reducing one triangle from each $ext(K_4^-)$ and every other reducible triangle. This provides (up to isomorphism) a unique ancestor for each connected cubic graph that has reducible triangles.

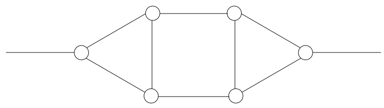

Figure 3: A subgraph $ext(K_4^-)$ with two reducible triangles that cannot be reduced at once.

We next identify the inverse operation of a bundled triangle reduction. For a graph $G = (V, E)$ call $S \subseteq V$ *extensible* if at least one vertex of every reducible triangle of G is contained in S. Then a *bundled triangle insertion* is to insert a triangle at each of the vertices in S. It is easy to see that this is the operation inverse to a bundled triangle reduction. After a bundled triangle insertion, all the $ext(K_4^-)$ subgraphs and other reducible triangles have been created in this last operation.

Applying bundled triangle insertion to distinct extensible sets of a graph G might yield isomorphic graphs. To avoid this, we define an equivalence relation on the extensible sets and will apply bundled triangle insertion to only one extensible set in every equivalence class.

The equivalence relation $\equiv$ on extensible sets is generated by the following two equivalences:

(a) If there is an automorphism γ of G with $\gamma(S) = S'$, then $S \equiv S'$.

(b) If $|S| = |S'|$ and $(S \setminus S') \cup (S' \setminus S)$ is the set of extremal vertices of a K_4^- in G so that each of S, S' contains exactly one vertex in this K_4^-, then $S \equiv S'$.

Lemma 3.1 *Consider a graph G and two extensible sets S, S' of G. Let $T(G, S)$, resp. $T(G, S')$, denote the graphs obtained by applying bundled triangle insertion to S, resp. S'. Then $T(G, S)$ and $T(G, S')$ are isomorphic if and only if $S \equiv S'$.*

Proof: If $S \equiv S'$ then for each of the generating relations an isomorphism can easily be constructed, so we will focus on the other direction.

Suppose that γ is an isomorphism from $T(G, S)$ to $T(G, S')$. Note that γ must map the set of subgraphs $ext(K_4^-)$ of $T(G, S)$ onto the set of subgraphs $ext(K_4^-)$ of $T(G, S')$, and also map the other reducible triangles of $T(G, S)$ onto the other reducible triangles of $T(G, S')$. Consider the mapping ϕ

from $V(T(G,S))$ to $V(G)$ defined by contracting the central two edges of each $ext(K_4^-)$ (drawn horizontally in Figure 3) and contracting all the edges of each reducible triangle that is not in an $ext(K_4^-)$. Define the mapping ϕ' from $V(T(G,S'))$ to $V(G)$ similarly. The permutation γ_0 of $V(G)$ defined by $\gamma_0(\phi(v)) = \phi'(\gamma(v))$ for all $v \in V(T(G,S))$ is an automorphism of G. Moreover, γ_0 maps S onto S' with the possible exception that when S contains one external vertex w, and no other vertex, of a K_4^- subgraph H, S' might contain either $\gamma_0(w)$ or the other external vertex of the K_4^- subgraph $\gamma_0(H)$. In either case, $S \equiv S'$ by the definition of equivalence. □

Since for every graph with reducible triangles the graph resulting from a bundled triangle reduction is uniquely determined, we get the following lemma.

Lemma 3.2 *If exactly one representative of every isomorphism class of cubic connected graphs up to $n-2$ vertices is given, then applying bundled triangle insertion to one member of each equivalence class of extensible sets that leads to a cubic connected graph on n vertices generates exactly one representative for every isomorphism class of cubic connected graphs on n vertices that contain reducible triangles.*

Thus no isomorphism rejection is needed for graphs with reducible triangles other than the equivalence relation on extensible sets.

One of the routines that can be time consuming in structure generation programs is the computation of automorphism groups. We used *nauty* [15] for this task. But though nauty is very efficient, the large number of calls can be expensive. The argument used in the proof of Lemma 3.1 shows that a non-trivial automorphism of $T(G,S)$ must derive from a non-trivial automorphism of G. If $\mathrm{Aut}(G)$ is trivial, this implies that $\mathrm{Aut}(T(G,S))$ is also trivial, thereby saving its computation (where $\mathrm{Aut}(G)$ stands for the automorphism group of G). This is one of main advantages of triangle insertion bundling. For 26 vertices about 78% of the graphs have a trivial group and the ratio is increasing. Even if $\mathrm{Aut}(G)$ is non-trivial, the orbits of $\mathrm{Aut}(G)$ can be used to choose vertex colours for $T(G,S)$ to speed up the computation of $\mathrm{Aut}(T(G,S))$ (see [15]).

3.2 Generating non-prime graphs without reducible triangles

In principle, the non-prime connected cubic graphs without reducible triangles on n vertices are generated by applying the edge insertion operation to each pair of edges in a graph on $n-2$ vertices that guarantees that no reducible triangles are present in the resulting graph. We call such a pair an extensible pair of edges. This may lead to the construction of isomorphic copies and we will now describe how we can make sure to list only pairwise non-isomorphic graphs.

The first task is to define, for each non-prime graph G without reducible triangles, a *canonical edge reduction* which is unique up to isomorphism. The result of performing the canonical edge reduction will be a graph G', uniquely determined by G up to isomorphism, from which G can be made by edge insertion. As is usual with the method of canonical construction path, we will only accept G if it is made from G' by the inverse of the canonical reduction; otherwise we will reject it.

To define the canonical edge reduction efficiently, we assign a 7-tuple $(x_0,\ldots,x_6)$ to every reducible edge $e = \{v,w\}$ and choose a reducible edge with the largest 7-tuple. The values of $x_0,\ldots,x_4$ are combinatorial invariants of increasing discriminating power and cost. The value of x_0 is 1 if e is part of a 4-gon and 0 otherwise, and x_1 is the negative of the number of vertices at distance at most 2 from e. The value of x_2 is equal to the number of 4-gons containing e, and x_3 is the negative of the number of vertices at distance at most 3 from e. Invariant x_4 is rather technical and gives only a small benefit, so we won't give the details. Each x_i is only computed if the previous values fail to choose a unique reducible edge and e is still potentially one of those with the greatest 7-tuple. In case there is more than one reducible edge with the greatest $(x_0, \ldots, x_4)$, we canonically label the graph using *nauty*, and define $x_5 > x_6$ such that $\{x_5, x_6\}$ is the lexicographically largest canonical labeling of an edge in the same orbit of $\mathrm{Aut}(G)$ as e. The discriminating power of $x_0, \ldots, x_4$ is enough that the more expensive computation of x_5, x_6 is only required in 8% of cases for $n = 26$, and this fraction is decreasing.

The values $x_0, \ldots, x_4$ are invariant under isomorphisms, so edges that are equivalent under the automorphism group have the same values. The values x_5, x_6 have an even stronger property: two edges are equivalent under the automorphism group if and only if x_5, x_6 are the same. This implies the same for the whole tuple $(x_0, \ldots, x_6)$: two edges have the same tuple if and only if they are in the same orbit of the automorphism group of the graph. Together with the definition of canonical labelling, this implies the following.

Lemma 3.3 *Let G_1 and G_2 be connected cubic graphs with reducible edges, and let γ be an isomorphism from G_1 to G_2. Let e_1 and e_2 be, respectively, reducible edges of G_1 and G_2 having greatest 7-tuples. Then $\gamma(e_1)$ is in the same orbit as e_2 under the action of $\mathrm{Aut}(G_2)$. Furthermore, reducing e_1 in G_1 produces a graph isomorphic to the result of reducing e_2 in G_2.*

The algorithm would work correctly if only x_5, x_6 are computed, but $x_0, \ldots, x_4$ are important for the efficiency of the algorithm. While for x_5, x_6 a canonical form must be computed, the earlier values are based on purely local criteria that are cheaper to compute. Furthermore these criteria allow some look-ahead. For example, when extending a graph G by inserting a new edge, it is easy to decide already on the level of G whether x_0 will be 1 or 0 for this edge in the extended graph $\bar{G}$ and whether other edges with $x_0 = 1$ in $\bar{G}$ will exist. This allows us to avoid the construction of a lot of children that would afterward be rejected.

What we still have to make sure is that from a graph with $n - 2$ vertices we never construct two isomorphic graphs that are both accepted. When applying the edge insertion operation to a graph G we first determine its automorphism group γ. After constructing the set of all extensible edge pairs, we compute the orbits of γ on the pairs and apply the edge insertion operation to exactly one pair in each orbit. This suffices to prevent isomorphic graphs from being accepted, as the following lemma shows. The fact that all graphs can still be constructed can be seen by observing that applying the operation to edge pairs in the same orbit gives isomorphic graphs.

Lemma 3.4 *If for one representative of each isomorphism class of cubic connected graphs with $n - 2$ vertices the edge insertion operation is applied to one edge pair in each orbit of extensible edge pairs, and a non-prime graph with n vertices and without reducible triangles is accepted if and only if the last edge inserted has a maximum value of $(x_0, \ldots, x_6)$, then exactly one representative of each isomorphism class of cubic connected non-prime graphs with n vertices and without reducible triangles is accepted.*

Proof: From Lemma 3.3 we know that isomorphic graphs must be made from the same parent. So assume that G is extended by applying edge insertion to edge pairs $p_1 = \{e_1, e_1'\}$ and $p_2 = \{e_2, e_2'\}$ in G and that the results are two isomorphic graphs G_1, G_2 that are both accepted. The inserted edges $\bar{e}_1$ and $\bar{e}_2$ must both have maximum $(x_0, \ldots, x_6)$ (else G_1 or G_2 will not be accepted), so by Lemma 3.3 there is an isomorphism from G_1 to G_2 that maps $\bar{e}_1$ to $\bar{e}_2$. But then restricting the isomorphism to the vertices that already belonged to G gives an automorphism of G mapping p_1 to p_2, showing that they were in the same orbit of the automorphism group of G, contrary to our procedure. □

Together, Lemma 3.2 and Lemma 3.4 give the following theorem.

Theorem 3.5 *When recursively applied, beginning with all prime graphs on up to n vertices, the algorithm just described constructs exactly one representative of every isomorphism class of cubic connected graphs on n vertices.*

3.3 Generating graphs with girth at least 4 or 5

The algorithm was developed and optimized for generation without a girth restriction, but it can be adapted to instead generate connected cubic graphs with a non-trivial lower bound on the girth. This is done by some simple look-aheads.

If G has at least 3 reducible triangles, then at least one of them remains when edge insertion is done. Since we favour triangle reduction over other edge reductions, the expanded graph will not be accepted. Therefore all the descendants of G will be made by triangle insertion and so will have triangles.

Similarly, if G has exactly 2 reducible triangles, but these are not connected by an edge, all the descendants of G will have triangles. This is because of the definition of x_0: any edge insertion that destroys both reducible triangles will give give an inserted edge that is not on a 4-cycle, but the expanded graph has reducible edges that are on 4-cycles.

In a search for cubic graphs with girth at least 4, none of those descendants need to be generated. By means of these look-aheads and others, the search tree can be substantially pruned so that girth bounds of 4 or 5 can be imposed quite efficiently. Also in these cases the program is faster than the ones described in [3] and [20]. It is also faster than the implementations reported in [24], which use operations that remain within the classes defined by the girth bounds, though it remains to be determined whether those algorithms can be implemented more efficiently.

This pruning technique is likely to become rather less efficient for larger lower bounds on the girth. For extreme girth, close to the maximum possible for a given number of vertices, the fastest method is that of McKay, Myrvold and Nadon [17].

4 Testing

The generator was used to generate all cubic graphs up to 32 vertices, with girth lower bounds of 3, 4 or 5. The numbers of graphs agreed with previously published numbers or numbers obtained by running minibaum.

The graph counts, running times and a comparison with minibaum are given in Table 2. Our generator is called *snarkhunter*.

As it would take more than 3 weeks to generate all cubic graphs with 30 vertices on a single CPU, we can split the generation into independent parts and execute them on multiple CPUs. This can be done by the method described in [16]: each CPU generates the whole search tree up to some level ℓ, then just its own portion of the tree beyond that level. Since unrelated branches of the tree are entirely independent, this is very easily implemented. For the generation of the cubic graphs with 30 and 32 vertices we used $\ell = 24$ and found that the overhead in paralellization was a tiny part of the total.

	All connected cubic graphs			
$\lvert V(G)\rvert$	# graphs	snarkhunter (sec)	minibaum (sec)	speedup
20	510 489	1.2	5.7	4.75
22	7 319 447	16	74	4.59
24	117 940 535	261	1 166	4.47
26	2 094 480 864	4 826	20 748	4.30
28	40 497 138 011	100 179	440 870	4.40
30	845 480 228 069	2 240 049		
32	18 941 522 184 590	53 177 371		

	Connected cubic graphs with girth at least 4			
$\lvert V(G)\rvert$	# graphs	snarkhunter (sec)	minibaum (sec)	speedup
20	97 546	0.8	2.5	3.13
22	1 435 720	11	34	2.99
24	23 780 814	191	542	2.85
26	432 757 568	3 626	10 107	2.79
28	8 542 471 494	76 218	216 837	2.84
30	181 492 137 812	1 756 557		
32	4 127 077 143 862	42 288 975		

	Connected cubic graphs with girth at least 5			
$\lvert V(G)\rvert$	# graphs	snarkhunter (sec)	minibaum (sec)	speedup
20	5 783	0.1	0.6	6.00
22	90 938	1.7	9.3	5.47
24	1 620 479	27	158	5.79
26	31 478 584	519	3 047	5.87
28	656 783 890	11 073	63 821	5.76
30	14 621 871 204	275 251		
32	345 975 648 562	6 473 440		

Table 2: Counts and generation times for classes of cubic graphs. Times are for C code compiled by gcc and run on an Intel Xeon L5520 CPU at 2.27 GHz. They include writing the graphs to a null device. As minibaum depends more strongly on memory performance than snarkhunter, the speedup in the last column depends strongly on the actual processor architecture. For other Intel Xeon processors it was considerably larger than given in the table.

5 Closing remarks

As a measure of how much extra improvement might be possible, we note that the mere act of copying the graphs to an output buffer (with no alteration to the internal structure except for inserting a few null characters), and writing them to the output, contributes 5–9% of the running time.

One of the main motivations for the new generator was the generation of *snarks* [12] – that is cyclically 4-connected cubic graphs that do not allow edge colourings with 3 colours. This subclass of cubic graphs is especially interesting as a source for possible counterexamples to graph theoretic conjectures. The fastest way to get complete lists so far was to use minibaum and filter the output (see [6]). The edge insertion operation allows a limited look-ahead detecting a lot of cases where the resulting graph will be edge 3-colourable. This results in a speed-up compared to previous methods that even exceeds the speed-up for the classes of graphs discussed here. Details will be published in another paper [4] together with some statistics on the snarks and counterexamples to published conjectures that were found by testing the snarks.

The latest version of the program can be downloaded from [5].

Acknowledgements

This work was carried out using the Stevin Supercomputer Infrastructure at Ghent University. Jan Goedgebeur is supported by a PhD grant from the Research Foundation of Flanders (FWO). Brendan McKay is supported by the Australian Research Council.

References

[1] R.E.L. Aldred, G. Brinkmann, D. Van Dyck, V. Fack, and B.D. McKay. Graph structural properties of non-yutsis graphs allowing fast recognition. *Discrete Applied Mathematics*, 157(2):377–386, 2009.

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

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

[4] G. Brinkmann, J. Goedgebeur, J. Hägglund, and K. Markström. Generation and properties of snarks. In preparation.

[5] G. Brinkmann, J. Goedgebeur, and B.D. McKay. Homepage of snarkhunter: `http://caagt.ugent.be/cubic/`.

[6] G. Brinkmann and E. Steffen. Snarks and reducibility. *Ars Combinatoria*, 50:292–296, 1998.

[7] 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.

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

[9] 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.

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

[11] 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.

[12] M. Gardner. Mathematical games: Snarks, boojums and other conjectures related to the four-color-map theorem. *Scientific American*, 234:126–130, 1976.

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

[14] 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.

[15] 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.

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

[17] B.D. McKay, 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.

[18] B.D. McKay and G.F. Royle. Constructing the cubic graphs on up to 20 vertices. *Ars Combinatoria*, 21a:129–140, 1986.

[19] B.D. McKay and N.C. Wormald. Automorphisms of random graphs with specified degrees. *Combinatorica*, 4:325–338, 1984.

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

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

[22] R.C. Read. Some enumeration problems in graph theory. Doctoral Thesis, *London University*, (1958).

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

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

[25] N.C. Wormald. Models of random regular graphs. In *Surveys in Combinatorics, 1999*, J.D. Lamb and D.A. Preece, eds, pp. 239–298.