Constructing the cubic graphs on up to 20 vertices.

Brendan D. McKay
Computer Science Department
Australian National University
Canberra
Australian Capital Territory

Gordon F. Royle
Department of Mathematics
University of Western Australia
Nedlands
Western Australia 6009

Abstract. This paper gives a detailed description of a method of constructing cubic graphs. All the non-isomorphic, connected cubic graphs on up to 20 vertices are found by this method, and catalogued with the graph theoretic properties of connectivity, order of automorphism group, chromatic number and index, diameter and girth, hamiltonicity and the size of a maximum independent set.

The cubic graphs on 16 and 18 vertices have previously been constructed by Faradzhev, but his catalogue is not readily available. To our knowledge this is the first construction of the cubic graphs on 20 vertices.

§0. Introduction.

In this section we describe the background to this research, and the terminology and notation to be used throughout this paper.

0.1 Basic Notation

Throughout this paper, we follow the graph theoretic notation of Behzad and Chartrand [2], except that a **graph** will be undirected without loops or multiple edges. In particular, a graph is regular if each vertex is adjacent to the same number of edges (that number being called the **valency**). A **cubic graph** is a regular graph of valency 3.

The following denote special classes of graphs. $\mathfrak{g}(v)$ denotes the regular graphs of valency v, $\mathfrak{x}$ denotes the graphs such that each vertex has valency 2 or 3, and $\mathfrak{f}$ denotes the graphs whose connected components are regular of valencies 2 or 3, such that at least one component is regular of valency 2, and at least one component is regular of valency 3.

Permuting the labels on the vertices of a graph produces an **isomorphic** graph. For each isomorphism class of graphs we distinguish one of the possible labellings, called the **canonical labelling**. The graph with this distinguished labelling is said to be canonically labelled. A canonical labelling algorithm is an algorithm that, when presented with a graph, produces the canonically labelled isomorph. A rigorous definition and description may be found in McKay [8].

0.2 The search for cubic graphs

Considerable effort has been expended in the production of catalogues of graphs of various kinds. Such catalogues have uses in suggesting conjectures, providing counterexamples or as sample graphs. As cubic graphs are of great importance in graph theory, several catalogues of cubic graphs have been produced. The cubic graphs on up to 10 vertices were drawn by Balaban [1]. Petrenjuk and Petrenjuk [10] found all the cubic graphs on up to 12 vertices and Faradzhev [7] found those on up to 18 vertices. Meanwhile Bussemaker, Cobeljic, Cvetkovic and Seidel [3],[4] produced a catalogue of all the cubic graphs with up to 14 vertices including information on several of their properties. As the listing of the graphs produced by Faradzhev is not readily available, we decided to repeat his work on 16 and 18 vertices, extend it to 20 vertices, and produce a catalogue similar to that in [3].

§1. Method

This section gives a detailed description of the method used for the construction of cubic graphs.

1.1 Definitions

For any graph **G** $\in \mathcal{H}$, we define an **ear of length** k, as a sequence $v_0, v_1, \ldots, v_k$ of vertices such that

(1) v_{i-1} and v_i are adjacent, for $1 \le i \le k$;
(2) $v_1, \ldots, v_{k-1}$ have valency 2, and v_0, v_k have valency 3;
(3) the v_i are distinct, except that $v_0 = v_k$ is permissible

The **internal vertices** of the ear are

$\{v_1, \ldots, v_{k-1}\}$ if $v_0 \neq v_k$, and
$\{v_0, \ldots, v_{k-1}\}$ if $v_0 = v_k$

and the **end vertices** are

$\{v_0, v_k\}$ if $v_0 \neq v_k$, and
$\{x \mid x$ is adjacent to v_0, $x \neq v_1, v_{k-1}\}$ if $v_0 = v_k$.

We can distinguish 3 different types of ear (see Figure 1)

(a) type 0 ear - $v_0 \neq v_k$, and v_0 is adjacent to v_k.
(b) type 1 ear - $v_0 \neq v_k$, and v_0 is not adjacent to v_k.
(c) type 2 ear - $v_0 = v_k$.

Notice that ears of type 0 or 1 have two end vertices, and ears of type 2 have only one end vertex.

The **quality** of a vertex is defined to be the number of vertices at distance 2 from that vertex plus 100 times the number of vertices at distance 3. Notice that this definition assigns an integer to each vertex of a graph in a fashion that is independent of the labelling. The purpose of such a definition is to provide a simple method of distinguishing the vertices of a graph to reduce uses of the canonical labelling algorithm as much as possible (see 1.2 below). Our particular definition is arbitrary.

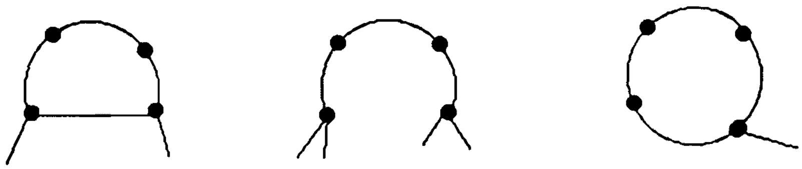

(a) a type 0 ear (b) a type 1 ear (c) a type 2 ear

Figure 1.

1.2 Parents and a forest

For any graph **G** ● $\mathcal{H}$, we define the **parent** of **G** to be the graph **H** $\in \mathcal{H}$ formed by applying the following rules.

(1) If the graph **G** ● $\mathfrak{G}(2)$ or **G** $\in \mathfrak{f}$, then **G** has no parents.

(2) If the graph **G** is cubic

From the set of vertices of lowest quality, select the one with the lowest canonical label, and form **H** by removing that vertex from **G**.

(3) If the graph **G** is not regular

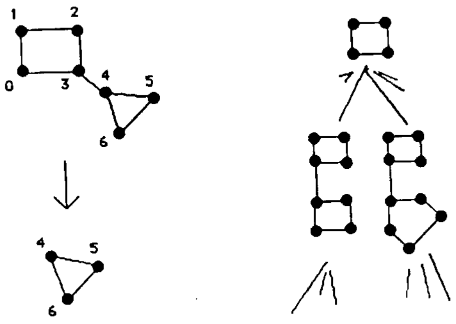
From the set of ears of lowest type, select the subset of those of greatest length. Then choose from this subset the ear containing the vertex with the lowest canonical label, and form **H** by removing all the internal vertices of that ear from **G**.

As each graph has at most one parent, and parents are always smaller than their children, the set $\mathcal{H}$ forms a forest of directed rooted trees under the parent relation. The roots of the trees are precisely the graphs $\mathfrak{g}(2) \cup \mathfrak{f}$, and the leaves (that is, nodes without children) are precisely the cubic graphs $\mathfrak{g}(3)$. The **reduction path** of a graph **G** is the path from **G** to the root of the tree it lies in (See Figure 2).

1.3 Searching the forest

It is clear that all the connected cubic graphs will be on the trees with roots that are regular of valency 2, and that all the disconnected cubic graphs will have roots in $\mathfrak{f}$. It is also clear that considering only nodes with less than N+1 vertices selects a finite subtree from each tree, which still contains the reduction paths for all the connected cubic graphs on N vertices.

Each of these finite subtrees is searched using a depth first search (also known as a backtrack algorithm).



(a) The parent forming operation

(b) A small portion of one of the trees

Figure 2.

1.4 Finding the children and pruning the tree

Due to the immense size of the trees, and the relative scarcity of the N-vertex cubic graphs, it is important to recognise as early as possible when a particular child cannot possibly lead to one of the **target nodes** (cubic graphs with N vertices).

A target node has N vertices, and 3N/2 edges. Hence, its parent will have N -1 vertices and 3N/2 - 3 edges (which follows immediately from rule (2) above). Any other node on the path to the root is formed by removing an ear from its immediate child. Removing any ear will remove m vertices and m +1 edges, for some $m \geq 1$, so we can find a bound for the number of edges that any node on the path from a target node to the root must satisfy.

If the node has n vertices and e edges, then the smallest value that e could take occurs when N-n-1 ears of length one have been removed (each ear consisting of one vertex and two edges), whilst the greatest value that e could take occurs when one ear of length N-n-1 has been removed.

Therefore at any stage we must have

$$e \geq 3N/2 - 3 - 2(N - n - 1),$$

that is $\quad e \geq 2n - N/2 - 1, \qquad (*)$

and $\quad e \leq 3N/2 - 3 - (N - n),$

that is $\quad e \leq N/2 + n - 3.$

Now consider how the children of a node are found. The children of a node are bigger than it, and due to the definition of 'parent', are formed by adding an ear to it (forming a non-cubic graph) or by adding a single vertex (forming a cubic graph). If the node **G** has N - 1 vertices and 3N/2 - 3 edges, then it may have a target node as a child. This is checked by adding one vertex and 3 edges, making a candidate child **C**, and then applying the parent rules to **C** to see whether **G** is in fact its parent. This final stage is achieved by seeing whether the newly added vertex is in the same orbit as the one that would be removed from **C**. (For removing the vertex that has just been added forms **G**). If the node **G** has N - 1 vertices but not the right number of edges, then it has no children leading to a target node and we backtrack.

We form the children of any other node in a similar fashion. A list of candidate children is formed by adding ears of all possible lengths, between all possible inequivalent pairs of points. The canonical labelling algorithm **nauty [9]** is used to find the orbits of points, and of pairs of points under the action of the automorphism group of the graph. Then only one pair of points from each orbit is used. This ensures that the candidate children are all different (see Proposition 1.5 below). Of course when adding ears of type 2 which have only one end vertex, we only use one point from each of the orbits on points. Then the parent rules are applied to each candidate child **C**, and it is accepted as a genuine child only if the ear just added is equivalent to the one that would be removed to form the parent of **C**.

We try to avoid using the canonical labelling algorithm as much as possible, due to its heavy use of computer time. For example if the newly added ear is the only one of lowest type, or the only one of greatest length amongst those of lowest type, then we are certain that we have a genuine child regardless of the canonical labelling.

We prune the tree by using the bounds on edges given above. If a node **G** has n vertices and e edges, then its child **C** will be formed by adding an ear, thus adding m vertices and m+1 edges, for some m. The equation (*) must still be satisfied after the addition of this ear, so

$$e + m + 1 \geq 2(n + m) - N/2 - 1,$$

that is $$m \leq e + 2 + N/2 - 2n.$$

The second bound may also be used to place a minimum length on the possible ears, but in practice such a bound is useless, as it is almost always less than 1.

1.5 Proposition

Each unlabelled connected cubic graph is found exactly once by the search procedure described above.

Proof

The argument above shows that only genuine children are produced, and that all such are produced, except those known not to be on the reduction path of any connected cubic graph on N vertices. Hence every connected cubic graph is found at least once. It remains to demonstrate that each child of a node is produced only once, that is, that distinct children of any given node are nonisomorphic. If the node has N - 1 vertices, then there is only one child produced. Now, suppose that two children of **G** are formed by adding ears **e** and **f** in non-equivalent places, but that **G+e** is isomorphic to **G+f**. Let g and h be isomorphisms between **G+e**, **G+f** and their (identical) canonically labelled isomorph **H**.

$$g: G+e \to H$$

$$h: G+f \to H$$

Then applying the parent rules to **H** will select an ear in the same orbit as g(**e**), and hence the same orbit as h(**f**). So g(**e**) must be in the same orbit as h(**f**). Therefore we can find an isomorphism

$$i: G+e \to G+f$$

such that i(**e**) = **f** by combining g, h^{-1} and an automorphism of **G+f** if necessary.

Now restricting i to acting only on the vertices of **G** produces an automorphism of **G**, such that the end vertices of **e** are mapped to the end vertices of **f**, which contradicts the fact that ears of a particular length are only added between non-equivalent pairs of points (or, in the case that **e** and **f** are type 2 ears, non-equivalent single points).¶

This algorithm falls into the class of **orderly algorithms** in the sense of Colbourn and Read **[5]**,**[6]** & **[11]**, because we apply a 'canonicity' test to each graph produced (in this case the test is whether the graph is a bona-fide child) and if it passes the test then we are sure that it is uniquely produced and do not need to do any duplicate checking.

1.6 Implementation

The algorithm described above was coded in the **C** programming language. It was executed in late 1984 on the VAX 11/750 machine at the Department of Psychology, University of Western Australia. The approximate times for generation were: 14 vertices - 10 min, 16 vertices - 2 hours, 18 vertices - 30 hours and 20 vertices - 500 hours.

§2. Results

The following numbers of non-isomorphic, connected, cubic graphs were constructed.

These numbers are in agreement with those found by Faradzhev **[7]**, and the theoretical number of connected cubic graphs according to Robinson and Wormald **[12]**.

Number of vertices	Number of graphs
4	1
6	2
8	5
10	19
12	85
14	509
16	4060
18	41301
20	510489

Standard algorithms were programmed for the properties of hamiltonicity, connectivity, diameter, girth, maximum independent set size, chromatic number and chromatic index.

This information together with the size of the automorphism group, which was found during construction, is tabulated below in the Appendix.

Our catalogue is stored on magnetic tape with each entry consisting of the graph, the orbits of its automorphism group, and the information listed above. The complete catalogue of all the cubic graphs from 4 - 20 vertices occupies 27 megabytes.

Property\Vertices =		4	6	8	10	12	14
Hamiltonian	Yes	1	2	5	17	80	474
	No				2	5	35
	1				1	4	29
Connectivity	2			1	4	24	139
	3	1	2	4	14	57	341
	1	1					
	2		2	2	1		
	3			3	15	34	34
Diameter	4				2	43	351
	5				1	6	93
	6					2	24
	7						6
	8						1

Property\Vertices =		4	6	8	10	12	14
Girth	3	1	1	3	13	63	399
	4		1	2	5	20	101
	5				1	2	8
	6						1
Chromatic number	2		1	1	2	5	13
	3		1	4	17	80	496
	4	1					
Chromatic index	3	1	2	5	17	80	475
	4				2	5	34
Maximum independent set size	1	1					
	2		1				
	3		1	4			
	4			1	17	7	
	5				2	73	80
	6					5	416
	7						13
Size of automorphism group	1					5	103
	2				2	22	159
	4			1	4	20	117
	6				2		4
	8				3	15	62
	12		1	1	1	4	7
	14						1
	16			2	2	9	35
	18					1	
	20				2		
	24	1				3	2
	28						2
	32				1	2	11
	36					1	
	48			1	1	2	1
	60				1		
	64					1	2
	72		1				
	96						1
	128						1
	336						1

Property\Graphs on		16 vertices	18 vertices	20 vertices
Hamiltonian	Yes	3841	39635	495991
	No	219	1666	14498
	1	186	1435	12671
Connectivity	2	1046	9398	101668
	3	2828	30468	396150
	3	14	1	1
	4	2167	12301	52272
	5	1499	22992	318071
	6	261	4400	109644
	7	101	1229	22603
Diameter	8	14	310	6206
	9	4	55	1352
	10		12	275
	11		1	59
	12			6
	3	3268	33496	412943
Girth	4	743	7350	91763
	5	48	450	5751
	6	1	5	32
Chromatic	2	38	149	703
number	3	4022	41152	509786
Chromatic	3	3848	39687	496430
index	4	212	1614	14059
	6	1074	21	
Maximum	7	2948	16183	635
independent	8	38	24948	268350
set size	9		149	240801
	10			703

Property\Graphs on		16 vertices	18 vertices	20 vertices
	1	1547	22124	327580
Size of	2	1261	11447	123116
automorphism	3	2	2	18
group	4	667	4633	39904
	6	15	29	49
	8	330	1934	13450
	10			7
	12	11	38	51
	16	147	733	4425
	18		3	
	20			6
	24	4	5	21
	30			1
	32	50	246	1358
	36		2	
	40			2
	48	6	9	9
	60			1
	64	14	72	376
	72		1	
	96	1	3	5
	120			1
	128	4	14	85
	216		1	
	240			1
	256		2	16
	288		1	
	320			2
	384	1	2	1
	512			2
	768			2

BIBLIOGRAPHY

[1] Balaban A.T. Valence-isomerism of cyclopolyenes. Rev. Roumaine Chim. 11(1966) 1097-1116; erratum 12(1967) 103.

[2] Behzad M.B, Chartrand C.D. Introduction to the theory of graphs. Allyn and Bacon. Boston (1971).

[3] Bussemaker F.C, Cobeljic M.S, Cvetkovic D.M, Seidel J.J. Computer Investigation of cubic graphs. Technological University Eindhoven, Dept of Math., Technical Report 76-WSK-01 (1976).

[4] Bussemaker F.C, Cobeljic M.S, Cvetkovic D.M, Seidel J.J. Cubic graphs on ≤ 14 vertices. J. Combinatorial Theory. Ser. B. 23 (1977) 234-235.

[5] Colbourn C.J, Read R.C. Orderly algorithms for graph generation. Intern. J. Computer Math. 1979, Sect. A. 7, 167-172.

[6] Colbourn C.J, Read R.C. Orderly algorithms for generating restricted classes of graphs. J. Graph Theory 3 (1979) 187-196.

[7] Faradzhev I.A. Constructive Enumeration of combinatorial objects. Problemes Combinatoires et Theorie des Graphes Colloque Internat. CNRS 260. CNRS Paris (1978) 131-135.

[8] McKay B.D. Practical graph isomorphism. Proceedings of the 10th Manitoba Conference on Numerical Maths and Computing. Congressus Numerantium, 30 (1981) 45-87.

[9] McKay B.D. nauty User's guide. Australian National University Computer Science Technical Report TR-CS-84-05 (1984).

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

[11] Read R.C. Every one a winner. Ann. Discrete Math. 2 (1978) 107-120.

[12] Robinson R.W, Wormald N.C. Numbers of cubic graphs. Journal of Graph Theory. Vol 7 (1983) 463-467.