Catalogs of Regular Graphs

Robert A. Beezer
John Riegsecker
Department of Mathematics and Computer Science
University of Puget Sound
Tacoma, Washington 98416

May 12, 1992

Abstract

Exhaustive catalogs of graphs can be an indispensable tool for computerized investigations in graph-theoretical research. We describe an algorithm for the creation of catalogs of regular graphs. Using this algorithm, all of the regular graphs on thirteen or fewer vertices have been created.

1 Introduction

Exhaustive catalogs of graphs can be an invaluable tool for research in graph theory. To be able to verify a conjecture quickly for all small graphs gives one extra confidence, and sometimes extra insight, into a problem. Or the discovery of a counterexample can lead to a necessary change in the hypothesis of a theorem. Additionally, in attempting to prove a result, one can use experiments on catalogs to help suggest or refine elements of a proof.

In this report we describe the generation of catalogs of regular graphs on thirteen or fewer vertices. The algorithm used is fairly simple, with the exception of the need for a routine to determine if two graphs are isomorphic. Basically, the algorithm works by creating a regular graph one vertex at a time. If a graph has k vertices of degree r, and the remainder of the vertices have degree less than r, then a vertex whose degree is less than r is chosen and edges are added so that this chosen vertex achieves degree r, while making sure no other vertex has its degree surpass r. If we start with a graph having n vertices and no edges, and apply this technique n times, the result will be a regular graph. By using this technique in an organized manner we will be able to create all regular graphs.

Here we describe the theoretical basis for our algorithm, in addition to some of the details of its implementation.

2 The Algorithm

We begin by describing the algorithm used to create regular graphs. We will let N_n be the graph on n vertices that has no edges. For a graph G, the vertex set will be denoted by VG

[1]This paper has been published in the *International Journal of Computer Mathematics* (51 (1994) 1-5). It is distributed in this form by the authors with the permission of the publisher. The theorems are numbered differently in this version, so please use the published copy for any citations of individual results.

and the edge set will be denoted by EG. The degree of the vertex v in the graph G will be written as $d(v,G)$. On several occasions we will construct sets of graphs; in each set we will consider isomorphic graphs as identical.

Definition 2.1 *Given integers n and r such that $0 \le r \le n-1$ define sets of graphs on n vertices, L_i, $i \ge 0$ by the following algorithm.*

Algorithm 2.2

```
Input(n, r)
L_0 ← {N_n}
for i ← 1 to ∞ do
  L_i ← ∅
  for each G ∈ L_{i−1} do
    S ← {v | v ∈ VG, d(v,G) < r}
    if S = ∅ then
      L_i ← L_i ∪ {G}
    else
      w ← v for some v ∈ S
      S ← S \ {w}
      t ← r − d(w,G)
      for all W = {w_1, w_2, ..., w_t} such that W ⊂ S do
        M ← G
        EM ← EM ∪ {{w, w_j} | w_j ∈ W}
        L_i ← L_i ∪ {M}
        end for
      end if
    end for
  end for
end
```

We will show that the set L_n is precisely the set of all regular graphs of degree r on n vertices, which we will denote by $R(n,r)$. The key feature of this algorithm is the choice of the vertex w. If we let w range over *all* vertices from S, then this algorithm would border on a brute force approach. However, we will prove that it suffices to choose *any* single vertex from S, thus making the algorithm non-deterministic.

Lemma 2.3 *$L_n \subseteq R(n,r)$.*

Proof Suppose that $G \in L_i$. Then it should be clear that G has at least i vertices of degree r when $0 \le i \le n$. Thus, if $G \in L_n$, then every vertex of G has degree r and G is a regular graph of degree r. ■

We wish to also show the opposite inclusion, $R(n,r) \subseteq L_n$. To this end, we make the following definitions. If v is a vertex of G, define $D(G,v)$ to be the graph formed from G by removing every edge incident to v. That is, $E(D(G,v)) = EG \setminus \{\{v,w\} \mid \{v,w\} \in EG\}$. Then define sets K_i recursively by

$$K_0 = R(n,r) \cup \{N_n\}$$

$$K_i = \{D(G,v) \mid G \in K_{i-1},\ v \in VG,\ d(G,v) \neq 0\} \cup \{N_n\}.$$

Notice that $K_n = \{N_n\}$. Finally, the notation $G = H \uplus K$ will mean that the graphs H and K have the same vertex set as G, and $EG = EH \cup EK$, while $EH \cap EK = \emptyset$.

Lemma 2.4 *Let L_i and K_i be as defined above, and suppose $G \in R(n,r)$. Then for every i there exists graphs $H_i \in L_i$ and $H'_i \in K_i$ such that $G = H_i \uplus H'_i$.*

Proof We will proceed by induction on i. $G \in R(n,r) \subset K_0$, and $L_0 = \{N_n\}$, so we can write $G = N_n \uplus G$ and satisfy the lemma for $i = 0$.

Assume that the lemma is true for $i = j-1$. Then we can write $G = H_{j-1} \uplus H'_{j-1}$, where $H_{j-1} \in L_{j-1}$, $H'_{j-1} \in K_{j-1}$. Now consider the construction of L_j from L_{j-1} at the stage where we have chosen H_{j-1} from L_{j-1} and constructed S, the set of vertices in H_{j-1} with degree less than r.

If $S = \emptyset$, then the algorithm adds H_{j-1} to L_j, so $H_{j-1} \in L_j$. Also, if $S = \emptyset$, then H_{j-1} must be regular of degree r, so $H'_{j-1} = N_n$. Thus,

$$G = H_{j-1} \uplus H'_{j-1} = H_{j-1} \uplus N_n$$

establishes the lemma for $i = j$ in the case where $S = \emptyset$.

Suppose now that $S \neq \emptyset$, and let w be the vertex of H_{j-1} chosen from S. Because $G = H_{j-1} \uplus H'_{j-1}$,

$$d(v, H_{j-1}) + d(v, H'_{j-1}) = r$$

for any vertex v. Then, since $d(w, H_{j-i}) < r$, $d(w, H'_{j-i}) > 0$. Let $H'_j = D(H'_{j-1}, w)$, which can easily be seen to be in K_j. If $\{v, w\} \in EH'_{j-1}$, then $d(v, H'_{j-1}) > 0$, so $d(v, H_{j-1}) < r$ and thus $v \in S$. Every edge of H_{j-1} has at least one endpoint of degree r, so if $v \in S$, $\{v, w\} \notin EH_{j-1}$. Create H_j by setting $EH_j = EH_{j-1} \cup \{\{v, w\} \mid \{v, w\} \in EH'_{j-1}\}$. Because each new edge of H_j joins w to some vertex from S, and $d(w, H_j) = r$, it should be clear that H_j will be one of the graphs M created by Algorithm 2.2, so $H_j \in L_j$. Since the edges removed from H'_{j-1} in the formation of H'_j are precisely the edges added to H_{j-1} in the formation of H_j, we have $G = H_j \uplus H'_j$. This establishes the lemma for $i = j$ in the case where $S \neq \emptyset$. ■

Lemma 2.5 $R(n,r) \subseteq L_n$.

Proof Suppose that $G \in R(n,r)$. Then, applying Lemma 2.4 with $i = n$ yields the existence of graphs $H_n \in L_n$ and $H'_n \in K_n$ such that $G = H_n \uplus H'_n$. However, since $K_n = \{N_n\}$, $H'_n = N_n$ and therefore $G = H_n \in L_n$. ■

Combining Lemma 2.3 and Lemma 2.5 yields our main result, which shows that we can use Algorithm 2.2 to construct $R(n,r)$ by halting after the construction of L_n.

Theorem 2.6 $L_n = R(n,r)$.

3 Practical Matters

In this section we describe some of the techniques used to implement Algorithm 2.2. A catalog composed of regular graphs of degree r on n vertices will be referred to as an (r,n)-catalog.

The algorithm allows for the vertex w to be any element of S. In practice, it appears most efficient to choose w to be a vertex of minimum degree in the graph taken from L_{i-1}.

In this way, the maximum number of new edges are added to the graph, and the regular graphs are formed as quickly as possible. Usually, $L_i = L_n$ when i is about 3 or 4 less than n. An experiment was performed using the creation of the (4, 11)-catalog to analyze the effect of choosing w as the vertex of maximum degree in S. Using the maximum degree vertex required 23% more time than using a vertex of minimum degree, and over 2.5 times as much memory.

After adding edges to a graph from L_{i-1}, the newly constructed graph is added to L_i, provided that an isomorphic copy does not already exist in the set. Thus we need to determine when two graphs are isomorphic. This was handled by using a canonical labeling algorithm created by B.D. McKay. This algorithm was coded by the first author in PASCAL from the FORTRAN description given in [2]. However, McKay also makes available more efficient source code in C and a user's manual [3]. A graph input to this routine will be returned as an isomorphic graph, but with a different labeling. If two graphs are isomorphic, and both are relabeled by this routine, then they will be *equal* graphs — that is, the identity mapping is an isomorphism.

Thus, when a graph is added to a set, we use the canonically labeled form. When a new graph is created, it is first canonically labeled. Then we can easily determine if the new graph is already in the set by simply testing the equality of the representations of the new graph and the current elements of the set.

Isomorphism testing is a notoriously difficult (and time-consuming) problem and in this case it is far and away the most time-intensive part of the algorithm. However, McKay's backtracking algorithm is very efficient, and on the small graphs involved in generating catalogs it works quickly. The search tree created by the backtracking canonical labeling procedure neither gets too deep nor too wide.

4 Results

The catalogs for $n \leq 11$ can be easily run on a fast PC. For example, the construction of the (4, 11) catalog, which is the most time consuming, takes 23 minutes on a 25 MHz 486 IBM-compatible. For the catalogs on 12 and 13 vertices a Hewlett-Packard Series 400 workstation was used, a machine whose integer performance is rated at 12 MIPS. The (4, 13) catalog took 22 hours on this machine. However, the (6, 13) catalog, presented special problems. In this case, L_7 occupied about 14 megabytes of random access memory, and it appeared that L_8 would occupy about 56 megabytes. Thus, L_7, was broken into 8 disjoint sets, and the algorithm was run in parallel on these 8 sets. This resulted in much redundancy across the 8 different versions of each L_i, which could not be eliminated until the sizes became manageable within the memory available. This also resulted in using about 800 hours of time to complete the catalog.

Table 1 lists the the number of connected regular graphs generated for each catalog. (The total number of regular graphs, including disconnected ones, in each catalog can be easily infered from this table.) These figures were checked for $n \leq 9$ by creating catalogs of all the graphs with 9 or fewer vertices, and searching out the regular graphs from these larger catalogs. The numbers for the connected, trivalent case also coincide with the size of catalogs created by D.R. Guichard [4] using entirely different techniques that apply only to creating connected trivalent graphs.

Vertices	Degree											
	1	2	3	4	5	6	7	8	9	10	11	12
2	1											
3		1										
4	0	1	1									
5		1		1								
6	0	1	2	1	1							
7		1		2		1						
8	0	1	5	6	3	1	1					
9		1		16		4		1				
10	0	1	19	59	60	21	5	1	1			
11		1		265		266		6		1		
12	0	1	85	1 544	7 848	7 849	1 547	94	9	1	1	
13		1		10 778		367 860		10 786		10		1

Table 1: Connected Regular Graphs

Given a fixed number of vertices, the catalogs where the degree is equal to approximately half the number of vertices are the largest and most time consuming to compute. Our goal here was to create complete catalogs for all the possible degrees on fixed numbers of vertices. Thus, given the difficulty and time involved in computing the catalog for $n = 13$, we stopped at this point. However, this algorithm should suffice to compute the catalogs for larger values of n, but only for the smaller degrees.

Acknowledgment: The authors would like to thank their system manager, Kathy Bodily, for her assistance in completing this project.

References

[1] Robert A. Beezer, Generating Catalogs of Regular Graphs, University of Puget Sound Department of Mathematics and Computer Science Technical Report 91-1, August 1991.

[2] B.D. McKay, Practical Graph Isomorphism, *Cong. Numer.* **30** (1981) 45–81.

[3] B.D. McKay, nauty User's Guide (Version 1.5), The Australian National University Department of Computer Science Technical Report TR-CS-90-02.

[4] D.R. Guichard, personal communication.