SMALL GRAPHS ARE RECONSTRUCTIBLE

Brendan D. McKay

Computer Science Department, Australian National University,
Canberra, ACT 0200, Australia
bdm@cs.anu.edu.au

Abstract

With the help of a novel computational technique, we show that graphs with up to 11 vertices are determined uniquely by their sets of vertex-deleted subgraphs, even if the set of subgraphs is reduced by isomorphism type. The same result holds for triangle-free graphs to 14 vertices, square-free graphs to 15 vertices and bipartite graphs to 15 vertices, as well as some other classes.

1. Introduction.

Given an undirected simple graph G, the *isomorph-reduced deck* $\mathcal{ID}(G)$ of G is a set containing one member of each isomorphism type of vertex-deleted subgraph of G. A strong form of the "reconstruction conjecture" is that G is uniquely determined by $\mathcal{ID}(G)$ if $|VG| \geq 4$ [4]. For surveys of the graph reconstruction problem, we refer the reader to [1, 2, 5].

Although it seems unlikely that a counterexample would be small, we believe that testing this supposition is a useful step. Verification for up to 9 vertices was carried out by us almost 20 years ago [6], but to our knowledge no previous verification on 10 vertices has been made despite the graphs being available since 1985 [3]. No doubt this is due to the large number (over 12 million) of such graphs, which causes a nontrivial problem of data management. The algorithmic challenge is to reduce the number of *pairs* of graphs which need to be compared. We solve this problem by modifying an existing algorithm for graph generation in such a way that any pair of graphs forming a counterexample would be generated close together. This is sufficiently successful that we can verify the conjecture for over 3×10^9 small graphs, including all the graphs with up to 11 vertices.

2. The algorithm.

In [8], we presented a very general technique for generating families of combinatorial objects without isomorphs. We begin by describing this method in our limited context. For $n \geq 1$, let $\mathcal{G}_n$ denote the set of all labelled simple graphs with vertex-set $\{1, 2, \ldots, n\}$. Let S_n denote the symmetric group, and $\mathrm{Aut}(G)$ be the automorphism group of G, both as permutation groups acting on $\{1, 2, \ldots, n\}$.

The construction process relies on a function $m(G)$, whose value is an orbit of $\mathrm{Aut}(G)$. The important necessary property of $m(G)$ is that it be invariant under relabelling of the argument. Technically: for $G \in \mathcal{G}_n$ and $\phi \in S_n$, we must have $m(G^\phi) = m(G)^\phi$.

Armed with m, we can generate nonisomorphic graphs. If $W \subseteq V(G)$, let $G[W]v$ denote the graph formed from G by appending a new vertex v and adding all possible edges between v and W.

```
procedure generate($G$ : labelled graph; $n$ : integer)
    if $|V(G)| = n$ then
        output $G$
    else
        for each orbit $A$ of the action of $\mathrm{Aut}(G)$ on $2^{V(G)}$ do
            select any $W \in A$ and form $G' = G[W]v$
            if $v \in m(G')$ then
                generate($G', n$)
            endif
        endfor
    endif
endprocedure
```

Theorem 1 [8]. *For any $n \geq 1$, the call* generate(K_1, n) *will cause the output of exactly one graph from each isomorphism class of graphs of order n.* ∎

The recursive structure of generate defines a rooted tree whose nodes are the isomorphism types of graphs, and whose root is K_1. This lets us call one node the "parent" or "child" of another in the usual manner. In the notation of the algorithm, the isomorphism class of G is the parent of the isomorphism class of G'.

The nontrivial requirements of generate are seen to be the computation of $\mathrm{Aut}(G)$ and $m(G')$. Details of how this can be done efficiently using the author's program nauty [7] are given in [8].

For our current purposes, however, we choose $m(G')$ quite differently. Starting with any total ordering T of unlabelled graphs, define $m(G')$ in any manner such that the previous requirements are met and, moreover, for $v \in m(G')$, $G' - v$ is maximal amongst the vertex-deleted subgraphs of G'. This additional restriction on $m(G')$ has an important consequence.

Theorem 2. *Suppose G_1 and G_2 are two distinct graphs of order n having $\mathcal{ID}(G_1) = \mathcal{ID}(G_2)$. Then G_1 and G_2 have the same parent in* generate.

Proof. Our definition of m, and the structure of generate, ensure that the parent of the isomorphism type of G_1 is the isomorphism type of $G_1 - v_1$, where v_1 is chosen to make this subgraph maximal under T. Similarly for G_2 and $G_2 - v_2$. However, if $\mathcal{ID}(G_1) = \mathcal{ID}(G_2)$, we must have that $G_1 - v_1$ and $G_2 - v_2$ are isomorphic. ∎

The computational method should now be clear. We apply `generate` to construct the graphs with n vertices. Comparison of their isomorph-reduced decks is carried out within the set of children of each graph of order $n-1$.

The process we actually applied in our computations was as follows. The ordering T was chosen to favour fewer edges, then a more complicated function f of the degrees, then finally a definitive ordering produced by `nauty`. This definition allows us to compute $m(G')$ in phases for efficiency. First we find the vertices of maximum degree, then if there is more than one we find those maximising $f(G'-v)$. Nearly always that leaves a single vertex v and we take $m(G') = \{v\}$. If not, we complete the computation of $m(G')$ using `nauty`. Note that there may be more than one orbit of vertices v for which $G'-v$ is maximal under T, due to pseudosimilarity; we must select one of them to meet the rules stated above.

In our computations, the sets of children of each node numbered at most a few hundred (usually much less). Within these small sets, we compared isomorph-reduced decks using some invariants then, in the rare surviving cases, using `nauty`.

Instead of considering all graphs, we can restrict attention to some subclasses defined by a hereditary property. For example, if `generate` is modified to ignore those graphs $G[W]v$ which contain a triangle C_3, the result is isomorph-free generation of triangle-free graphs. We also considered graphs not containing squares C_4, and bipartite graphs. Finally, we considered graphs with maximum degree at most 5. All of these properties can be easily seen to be determinable from $\mathcal{ID}(G)$, so it is valid to restrict the exploration to within each subclass.

We conclude with a summary of our results.

Theorem 3. *The following classes of graphs are uniquely determined (within the set of all graphs) by their isomorph-reduced decks:*
(a) graphs of order 4–11;
(b) graphs of order 12 and maximum degree at most 5;
(c) triangle-free graphs of order 4–14;
(d) square-free graphs of order 4–15;
(e) bipartite graphs of order 4-15;
(f) bipartite graphs of order 16 and maximum degree at most 5. ∎

For the record, the number of graphs in each of the classes listed above is respectively 1 031 291 291, 495 369 040, 490 050 267, 116 180 700, 648 650 952, and 1 507 524 197. The total cpu time used, on a mixture of Sun workstations, was slightly less than one year.

I wish to thank Mark Ellingham for some useful comments.

References.

[1] J. A. Bondy, A graph reconstructor's manual, in "Surveys in Combinatorics, 1991", (Guildford, 1991), London Math. Soc. Lecture Note Ser. 166, Cambridge Univ. Press, Cambridge (1991) 221–252.

[2] J. A. Bondy and R. L. Hemminger, Graph reconstruction—a survey, *J. Graph Theory*, **1** (1977) 227–268.

[3] R. D. Cameron, C. J. Colbourn, R. C. Read and N. C. Wormald, Cataloguing the graphs on 10 vertices, *J. Graph Theory*, **9** (1985) 551–562.

[4] F. Harary, On the reconstruction of a graph from a collection of subgraphs, in "Theory of graphs and its applications", Academic Press, NY (1964) 47–52.

[5] B. Manvel, Reconstruction of graphs: progress and prospects, 250th Anniversary Conference on Graph Theory (Fort Wayne, IN, 1986), *Congressus Numerantium* **63** (1988) 177–187.

[6] B. D. McKay, Computer reconstruction of small graphs, *J. Graph Theory*, **1** (1977) 281–283.

[7] B. D. McKay, nauty User's Guide (version 1.5), *Tech. Rpt. TR-CS-90-02, Dept. Computer Science, Austral. Nat. Univ.* (1990).

[8] B. D. McKay, Isomorph-free exhaustive generation, submitted.

(Received 16/4/96)