On Isomorphism Testing in 3-Regular Graphs

W.L. Kocay
Department of Computer Science
University of Manitoba
Winnipeg, Manitoba
CANADA R3T 2N2

Abstract

The polynomial algorithms for isomorphism testing in 3-regular graphs known to date use set-wise stabilisation in 2-groups acting on singletons, pairs, and sometimes triples of vertices. In this note we describe a new, simpler way of "getting rid of the triples". Although the order of the complexity of isomorphism testing remains $O(n^3 \log n)$, the resulting algorithm is more efficient, since this portion of the set-wise stabilisation in the algorithm will be faster.

JCMCC 1(1987), pp. 131–140

1. Introduction.

We shall use the graph-theoretic terminology of Bondy and Murty [1], so that a graph X has vertex set V(X) and edge set E(X). If $A \subseteq V(X)$, then X[A] denotes the subgraph induced by A. [A,B] denotes the set of edges of X with one end in A and one end in $B \subseteq V(X)$.

Let X be a 3-regular graph. Choose an edge $e \in E(X)$. The polynomial graph isomorphism algorithms of Hoffman, Luks et al. [1,2,3,4] for 3-regular graphs find $Aut_e(X)$, the subgroup of Aut(X) which fixes the edge e. The technique depends on finding set-wise stabilisers in 2-groups. Subdivide e with a new vertex v_0, and find the distance partition of V(X) into $\{v_0\} + V_1 + V_2 + ... + V_h$, as indicated in Fig. 1. X is decomposed into a sequence of graphs $X_0, X_1, X_2, ..., X_{h+1}$, where $V(X_k) = \{v_0\} + V_1 + V_2 + ... + V_k$, and $E(X_k) = X[\{v_0\} + V_1 + ... + V_{k-1}] \cup [V_{k-1}, V_k]$. $G_k = Aut(X_k)$, for each k=0, 1, ..., h+1, so that $G_{h+1} = Aut_e(X)$. The algorithm successively finds G_{k+1} from G_k. This is done as follows.

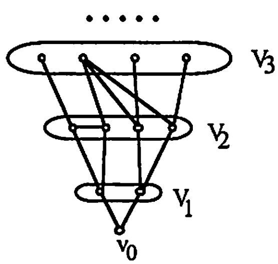

Fig. 1

Let $\phi_k: G_{k+1} \rightarrow G_k$ be the natural homomorphism. $Ker(\phi_k)$ fixes each vertex of X_k. It is very easy to find because of the 3-regularity of X. $K_{k+1} \equiv G_{k+1}/Ker(\phi_k)$ consists of all automorphisms of X_{k+1} which can be extended from G_k. In order to find K_{k+1}, it is necessary to consider the action of G_k on the edges $[V_k, V_k]$ and $[V_k, V_{k+1}]$ in going from X_k to X_{k+1}. This is done by finding set-wise stabilisers in G_k, which is a 2-group. Together, $Ker(\phi_k)$ and K_{k+1} define generators for G_{k+1}.

Vertices $x \in V_{k+1}$ which are joined to three vertices in V_k present a slight problem. In the algorithm of [3], G_k was allowed to act on $V_k + \binom{V_k}{2} + \binom{V_k}{3}$, the set of all singletons, pairs, and triples of vertices. In [2], the triples are eliminated by replacing each such $x \in V_{k+1}$ by a triangle, as shown in Fig. 2. This gives a new graph $\overline{X}_{k+1}$ which is still 3-regular, but which has no such triples. However, it may require the addition of a substantial number of new vertices and edges to X.

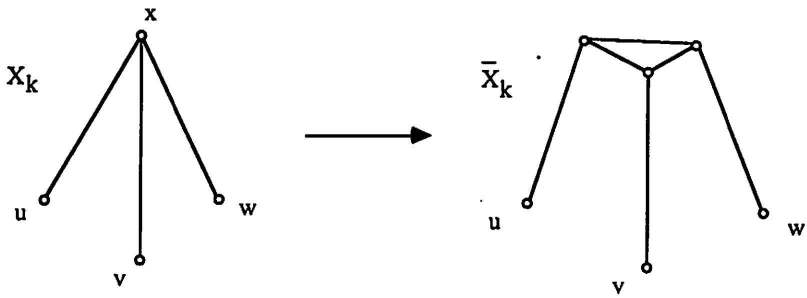

Fig. 2

In this note, we describe a simpler method of eliminating the triples. It does not change the complexity of the algorithm, which remains $O(n^3 \log n)$, but it will improve some of the set-wise stabilisation, giving a faster algorithm.

2. Triangles and Octahedra.

What one would like to do is to replace each $x \in V_{k+1}$ joined to $u,v,w \in V_k$ by three new edges uv, vw, wu in X_k, and colour them red, say, to distinguish them from the other edges of X_k. We would then want the set-wise stabiliser of the red edges to induce corresponding permutations of the vertices $x \in V_{k+1}$ so joined. The problem is that although each such $x \in V_{k+1}$ defines a red triangle in X_k, a red triangle in X_k need not necessarily correspond to a vertex in V_{k+1}. Call each red triangle of X_k a true triangle if there is a corresponding vertex in V_{k+1} and a false triangle if there is no corresponding vertex. Problems only arise if some element of G_k maps a false triangle to a true triangle. We show that this happens only in exceptional circumstances, which can be easily detected and dealt with.

2.1 Definition. For each $x \in V_{k+1}$ joined to three vertices $u,v,w \in V_k$, add three new edges uv, vw, and wu to $X[V_k]$, and colour them red, to distinguish them from other edges of X_k. $X[V_k]$ may now contain pairs of vertices u,v joined by two parallel red edges. In this case, replace the pair of red edges by a single edge, coloured double red, a new colour. We denote the two classes of red edges by R and RR, respectively, and collectively refer to both as red edges. R_k denotes the subgraph of $X[V_k]$ induced by the red edges (R and RR).

R_k consists of a number of triangles, some of which will be true triangles, and some of which will be false triangles. $G_k(R_k)$ denotes the subgroup of G_k which stabilises the edges R_k set-wise. We state a number of simple properties as lemmas.

2.2 Lemma. Each edge of R_k belongs to at least one true triangle. □

2.3 Lemma. If a triangle of R_k contains two RR edges, then the third edge is also RR.
Proof. Each $u \in V_k$ can be joined to at most two vertices of V_{k+1}. □

2.4 Lemma. Any triangle containing an RR edge is a true triangle.
Proof. Each $u \in V_k$ can be joined to at most two vertices of V_{k+1}. □

Suppose now that T=uvw is a false triangle of R_k. Each side of T belongs to a different true triangle, for otherwise T would be a true triangle. Let T_{uv}=uvx be the (unique) true triangle containing uv. Let T_{uw}=uwy be the true triangle containing uw. If x=y, then ux must be colored RR, as indicated in Fig. 3. In this case T is said to be a false triangle of type I, viz., one or more vertices of T is incident on an RR edge.

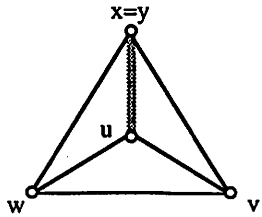

Fig. 3

If x≠y the situation of Fig. 4 holds. Here T_{vw}=vwz is the true triangle containing vw; and x,y, and z are three distinct vertices, for otherwise T would reduce to type I. In this case T is said to be of type II, viz., all vertices of T are incident on R edges, only.

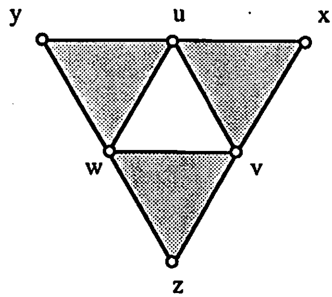

Fig. 4

An interesting case occurs when xyz also form a true triangle, as indicated in Fig. 5. In this case each of u,v,w,x,y, and z are incident on four R edges, so that each is adjacent to two vertices of V_{k+1}. This forms a connected component C of R_k. C is isomorphic to the graph of the octahedron. The triangles of C are alternately true and false, as indicated by the shading of Fig. 5. We call any such component of R_k an <u>octahedron</u>.

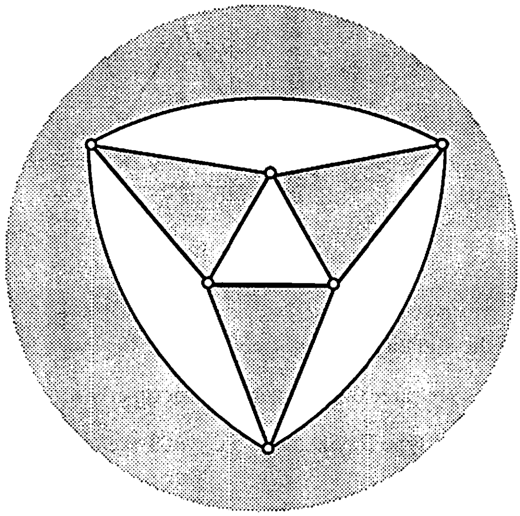

Fig. 5

2.5 Lemma. No false triangle of type I can be mapped by $G_k(R_k)$ to a true triangle.

Proof. Let T=uvw be a true triangle onto which a false triangle of type I has been mapped. Then vertex u, say, is incident on an RR edge ux (see Fig. 3). Since ux is an RR edge, it is contained in two true triangles. This forces u to be adjacent to at least 3 vertices of V_{k+1}, which is impossible.□

2.6 Theorem. A false triangle of type II can be mapped by $G_k(R_k)$ to a true triangle uvw only if the connected component of R_k containing uvw is an octahedron.

Proof. Let T=uvw be a true triangle onto which a false triangle of type II has been mapped. Let x,y, and z be the other vertices associated with a type II triangle, as indicated in Fig. 4. Since T=uvw is a true triangle, triangles uvx, uwy, and vwz are false, as is indicated by the shading. Let vxt be the true triangle containing vx. If t≠z, then v must be adjacent to three vertices of V_{k+1}, since vz is also contained in a true triangle. It follows that vxz froms a true triangle. Similarly wyz and uxy also from true triangles, so that the component of R_k containing T is an octahedron. □

So if R_k contains no octahedra, $G_k(R_k)$ necessarily maps true triangles to true triangles only. If R_k contains octahedra, then octahedra are mapped to octahedra, since they are connected components of R_k. If C_i and C_j are two octahedra such that C_i is mapped to C_j, then either the true triangles of C_i are mapped to the true triangles of C_j, or else all true triangles of C_i are mapped to false triangles of C_j.

The octahedra of R_k are very easy to find. Each vertex of V_k has degree at most 4 in R_k, so that we can find the components of R_k in at most $4.|V_k|$ steps. An octahedron is characterized as a 4-regular connected component with 6 vertices (and 12 edges). Suppose there are m octahedra $C_1, C_2, ..., C_m$. Perform the following steps.

```
Begin
  For i:=1 to m do begin
    create 2 new vertices T_i and F_i  { representing the true and false triangles of C_i }
  end
  Compute G_k(R_k)
  { now add the new vertices T_i and F_i, i=1,2,...,m  to V_k }
  For each generator g of G_k(R_k) do begin
    { we extend g to act on the T_i and F_i }
    For i:=1 to m do begin
      Choose a true triangle T of C_i  { this is easy to do since the true triangles of
            R_k are defined by vertices x∈ V_{k+1} initially }
      Compute g(T)
      Determine which C_j contains g(T)  { this will depend on the algorithm and
            data structures used to find the components of R_k }
      If g(T) is true then begin
        define g(T_i) := T_j
        define g(F_i) := F_j
      end
```

```
        else begin
            define g(T_i) := F_j
            define g(F_i) := T_j
        end
    end;
    { at this point, g(T_i) and g(F_i) are defined for all i }
  End; { all generators have been processed }
  Compute the set-wise stabiliser S_k of {T_1, T_2, ..., T_m} in G_k(R_k)
End;
```

The resultant set-wise stabiliser S_k is the group we want. Each element of S_k induces a permutation of the vertices $x \in V_{k+1}$ joined to three vertices of V_k. The subgroup of G_k which can be extended to X_{k+1} is a subgroup of S_k. The other vertices of V_{k+1}, those joined to one or two vertices of V_k, can be dealt with in the usual manner.

Consider the usual method (see [4]) used to find a set-wise stabiliser of a set M in a 2-group G acting on a set V_k. We first compute a tower of subgroups for G based on a tree of block systems for G and its subgroups. In the algorithm of [2], one tree is used for all groups encountered throughout the algorithm. At the bottom of the tree, the subgroups respecting the block partitions of V_k are point-wise stabilisers in G. We break G into cosets $H+\sigma H$, where the subgroup H fixes the first block partition in the tree. The stabiliser $C_M(G)$ is either $C_M(H)$, or $\langle C_M(H), \rho \rangle$, where $\rho \in \sigma H$ fixes M. Given σ, the algorithm consists of searching through the tree of block partitions to see whether the σ-coset of the corresponding subgroup contains such a ρ. The number of elements of G which must be examined in order to find $C_M(G)$ is at least $O(|V_k|)$ (if the recurrence of [4] is used, we get $O(|V_k|^2)$). Since we are really interested in the action of G on *pairs* of vertices, the quantity $\binom{V_k}{2} \approx V_k^2/2$ is the important one.

If R_k contains m octahedra, corresponding to 4m true triangles, or 12m edges (and at least 12m new vertices if the graph $\overline{X}_k$ of [2] is used, *not counting the triangles not* *contained in octahedra - i.e., probably most triangles*), then the present method requires introducing only 2m new vertices. They simply add another orbit to the partitions in the tree of block systems, and so do not affect the number of nodes it contains. The action of all the groups in the tower on the new vertices T_i and F_i must be computed, but this can be most easily done when T_i and F_i are being defined. Furthermore, the 4m vertices $x \in V_{k+1}$ corresponding to these octahedra all have degree three to V_k, so that they do not contribute to the next iteration, from G_{k+1} to G_{k+2}. If the other method is used there are *at least* 12m new vertices which must be added to V_{k+1}. Then $V_{k+1}^2/2$ increases by at least $12mV_{k+1} + 72m^2$. These new vertices (and pairs) must also be inserted in the tree of blocks partitions and this tends to increase the number of nodes in the tree.

So the use of octahedra should improve the efficiency of the set-wise stabilisation portion of the algorithm noticeably even though it will not change the order of complexity of the algorithm, which remains $O(n^3 \log n)$.

Acknowledgement. This research was supported by an operating grant from the Natural Sciences and Engineering Research Council of Canada.

References

1. M. Furst, J.E. Hopcroft, and E. Luks, A subexponential algorithm for trivalent graph isomorphism, 1980, Tech. Rept. 80-426, Dept. of Computer Science, Cornell University.
2. Zvi Galil, Christoff M. Hoffmann, Eugene M. Luks, Claus P. Schnorr, and Andreas Weber, An $O(n^3 \log n)$ deterministic and $O(n^3)$ probabilistic isomorphism test for trivalent graphs, Proc. 23rd IEEE Symp. on the Foundations of Comp. Sci., N.Y., 1982, pp. 118-125.
3. Christoff Hoffmann, Group Theoretic Algorithms and Graph Isomorphism, Lecture Notes in Computer Science #136, Springer-Verlag, New York, 1982.
4. Eugene M. Luks, Isomorphism of bounded valence can be tested in polynomial time, Proc. 21st IEEE Symp. on the Foundations of Comp. Sci., Rochester, N.Y., 1980, pp. 42-49.