MATCH
Communications in Mathematical and in Computer Chemistry

MATCH Commun. Math. Comput. Chem. **58** (2007) 431-443

ISSN 0340 - 6253

Groups & Graphs – Software for Graphs, Digraphs, and their Automorphism Groups

William Kocay*
Computer Science Department
St. Paul's College, University of Manitoba
Winnipeg, MB, Canada R3T 2N2
e-mail: bkocay@cc.umanitoba.ca

(Received June 29, 2006)

Abstract

This article is a brief description of the main features of the Groups & Graphs software package.

1. Introduction

Groups & Graphs is a software package for graphs and related discrete structures, and their automorphism groups. It began as software for graph editing and display, for computing automorphism groups, and comparing graphs for isomorphism. Today it includes algorithms for many aspects of graph theory, and related areas, including digraphs, graph embeddings on the plane, projective plane, sphere, and torus, as well as 3D polyhedra, projective configurations, and also fractals in the complex plane. It is useful for studying special graphs important in mathematical chemistry, such as fullerenes. Such molecular graphs can be drawn either on the plane, or on the surface of the sphere. Symmetric drawings of

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

graphs can often be automatically constructed. The number of Hamilton cycles can be counted, and maximum matchings can also be found. Kirchhoff or Laplacian matrices can be exported for finding eigenvalues with programs like Mathematica or Matlab. Groups & Graphs also includes many algorithms for permutation groups, with interaction between an automorphism group, and the graph/digraph it acts on.

The user interface is mostly menu, window, and mouse driven. Each graph is drawn in its own window. A number of graph editing tools are available. Groups & Graphs includes a number of algorithms for drawing graphs, including a "draw-symmetric" feature for finding drawings illustrating various symmetries, and also algorithms for drawing graphs on the plane, sphere, projective plane, and torus. Batch processing is also available, where files containing many thousands of graphs can be input for fast isomorphism testing. Graphs can be input either in G&G binary format, or in an ascii text format.

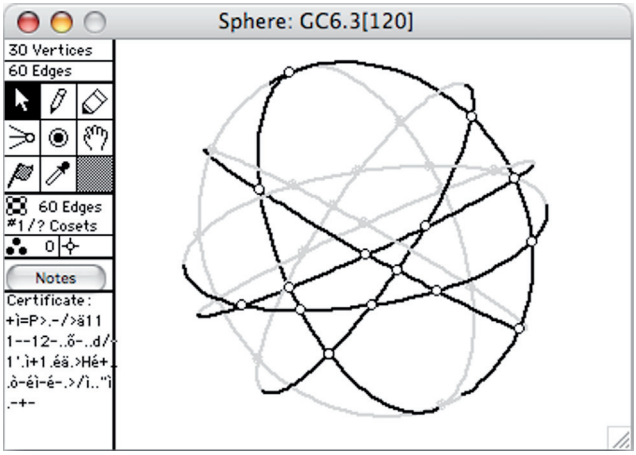

Figure 1, A graph drawn on the sphere

A brief description of the main algorithms in Groups & Graphs fol-

lows. The various tools for editing and constructing graphs and other mathematical objects are not described here, but descriptions can be found in the software, which can be downloaded from http://bkocay.cs.umanitoba.ca/G&G/G&G.html. Groups & Graphs currently runs on the OS X operating system. A GNU version for unix of the Hamilton cycle algorithm is also available.

2. Graph Algorithms

Automorphism Group

The automorphism group of a graph is computed by an algorithm described in Kocay [6]. Once the automorphism group of a graph has been computed, a *certificate* that identifies the isomorphism type of the graph is available. To compare large files of graphs for isomorphism, and to remove duplicates, it is only necessary to sort the file of certificates. The automorphism group is also used for finding symmetric drawings of a graph.

Hamilton Cycles

The "extended multi-path algorithm" is used to determine whether a graph is Hamiltonian. It is described in Kocay [9]. As HamCycle is an NP-Complete problem, there are graphs for which this algorithm must take a long time. These are usually non-Hamiltonian graphs. The algorithm is quite effective for most Hamiltonian graphs of reasonable size. It can be used to simply find a single Hamilton cycle, or to count the number of Hamilton cycles, or to save all Hamilton cycles to a file. It can also determine equivalence classes of Hamilton cycles, up to isomorphism.

Long Paths

If a graph is non-Hamiltonian, or too large for the HamCycle algorithm, an algorithm to find a long path is available, based on the

"crossover technique", described in Kocay and Li [7]. There are no guaranteed bounds on the length of a path that it will find, but it is extremely fast.

Planarity

The algorithm used to determine whether a given graph is planar is based on the Hopcroft-Tarjan algorithm [5,16]. Once a graph is found to be planar, its planar dual is constructed. A planar drawing can also be constructed.

Maximum Matchings

The Edmonds-Karp algorithm [13] is used to find a max-matching in a graph.

Line Graph, Inverse Line Graph, Graph Products

The line-graph and a number of graph products can be computed, including the direct product and lexicographic product of graphs. The inverse line graph can also be computed – if a graph G is the line graph of some graph H, then H can be constructed.

k-Factors

A k-factor of a graph G is a spanning subgraph of degree k. For example, a perfect matching is a 1-factor. The algorithm used to find a k-factor, where $k \geq 1$, is based on the technique of *balanced flows*, described in Kocay and Stone [8].

Subgraph Counts

Given a graph G, the numbers of subgraphs isomorphic to various small graphs are available (eg., triangles, quadrilaterals, other small cycles or small cliques).

Separating Sets

A *separating set* of a graph G is a set of vertices U such that $G-U$ is disconnected. This is also called a *vertex-cut*. A minimum separating set is found using a max-flow algorithm.

Graph Colouring

Two graph colouring algorithms are available – a breadth-first and a depth-first colouring. They are not optimal in any sense. If a graph is known to be planar, a 4-colouring algorithm is available. It is described in Carr and Kocay [2].

Draw Symmetric

Given a graph G, whose automorphism group $\mathrm{aut}(G)$ has been computed. An element $\sigma \in \mathrm{aut}(G)$ can be selected, and a drawing of G illustrating the symmetry σ can be constructed, when possible. The algorithm for doing this is described in Carr and Kocay [3]. Alternatively, a random $\sigma \in \mathrm{aut}(G)$ can be chosen, and a drawing constructed based on σ.

A number of other algorithms for graphs are available, which are not described here.

3. Digraph Algorithms

Several of the graph algorithms are also available for digraphs. Some algorithms specific to digraphs are also available.

Automorphism Group, Long Paths, Line Graphs, and Subgraph Counts

These graph functions are also available for digraphs. The subgraphs counts are available for a number of small digraphs, such as transitive tournaments, diamonds, and directed cycles.

Strong Components

A strong component of a digraph is a maximal subgraph with the property that there is a directed uv-path connecting any two vertices u, v in the subgraph. The algorithm used is the depth-first search algorithm of Tarjan [15].

Converse

The converse of a digraph is obtained by reversing the orientation of each edge.

4. Group Algorithms

All the groups constructed by Groups & Graphs are permutation groups. They are usually constructed as automorphism groups of discrete objects. A group is represented by a recursive data structure that is built using the Shreier-Sims algorithm [1,6]. Once a group G acting on a set X has been constructed, a number of subgroups and quotient groups can be found. If $\sigma \in G$ and $u \in X$, then u^σ denotes the image of u under the permutation σ. Permutations are multiplied from left to right, so that $\sigma\tau$ means "first σ, then τ". The orbit of u is represented by u^G. A group window shows the generators and orbits of G. Clicking in an orbit highlights the points of that orbit in the associated graph window. The subgroups of G which Groups & Graphs can construct are as follows.

Stabilizer

Given $u \in X$, the *stabilizer* of u is $G_u = \{\sigma \in G \mid u^\sigma = u\}$.

Centralizer

Given $\sigma \in G$, the *centralizer* of σ is $Z_\sigma = \{\tau \in G \mid \sigma\tau = \tau\sigma\}$.

Centre

The *centre* of G is $Z = \{\tau \in G \mid \sigma\tau = \tau\sigma, \forall\sigma \in G\}$.

Sylow Subgroups

Given a prime p, a *Sylow p-subgroup* of G is a subgroup of order p^k, where k is as large as possible.

Commutator Subgroup

Given $\sigma, \tau \in G$, the *commutator* of σ and τ is $[\sigma, \tau] = \sigma\tau\sigma^{-1}\tau^{-1}$. The commutator subgroup is generated by all the commutators of elements of G.

Once a subroup H of G has been constructed, the cosets of H in G can also be found. Given $\sigma \in G$, a *right coset* of H is a set $H\sigma = \{\tau\sigma \mid \tau \in H\}$. A number of coset graphs can be constructed, as well as quotient groups. The *kernels* associated with the quotient groups are also constructed.

Normalizer

Given a subgroup H of G, the *normalizer* of H is $N(H) = \{\sigma \in G \mid H\sigma = \sigma H\}$.

G mod H

The elements of G permute the cosets of H according to the rule $(H\sigma)^\tau = H\sigma\tau$. Thus, the elements of G induce a permutation of the cosets of H. The resulting group is denoted G mod H.

Orbit Constituent

Given an orbit u^G, the *orbit constituent* of G is the quotient group obtained by ignoring the other orbits of G.

Pair Group and Edge Group

Given $u, v \in X$ and $\sigma \in G$, $(u, v)^\sigma = (u^\sigma, v^\sigma)$ and $\{u, v\}^\sigma = \{u^\sigma, v^\sigma\}$. Thus G induces a group acting on ordered pairs (the *pair group* of G) and on unordered pairs (the *edge group* of G).

Symmetrize

This function is used for constructing graphs with a given group G acting as a group of symmetries. Given G acting on a set X, where $|X| = n$. Create a graph window containing n points, and add an edge $\{u, v\}$ to the graph. The Symmetrize command will construct the orbit $\{u, v\}^G$, and add all these edges to the graph. This can be used for constructing circulant graphs and other symmetric structures, such as block designs, via their bipartite incidence graphs.

Coset Graphs, Double Coset Graphs, Cayley Graphs
Given a subgroup H of G, a Schreier coset graph is formed by taking the right cosets of H as the vertex set, and edges determined by a set of generators of G mod H: given a generator τ of G, there is a directed edge connecting cosets $H\sigma$ and $H\sigma\tau$. If H is the identity subgroup, then the Schreier coset graph is a *Cayley graph*. A *double coset* of H is a set $H\sigma H$, where $\sigma \in G$. It is a union of one or more right cosets of H, say $H\sigma H = H\sigma_1 \cup H\sigma_2 \cup \ldots \cup H\sigma_k$. A double coset graph of H for the double coset $H\sigma H$ is formed by taking the right cosets of H as the vertex set. Each coset $H\tau$ is joined to $H\sigma_1\tau, H\sigma_2\tau, \ldots, H\sigma_k\tau$.

Block Systems
Given an orbit u^G of G, it is often possible to partition the orbit into sets $U_1, U_2, \ldots, U_k$ such that every $\sigma \in G$ induces a permutation of the U_i; that is, each U_i is mapped entirely to a U_j. Such a partition of the orbit is called a *block system*. Groups & Graphs can find a block system if one exists, and the quotient group acting on the blocks.

A number of other algorithms for groups are available, which are not described here.

5. Graph Embeddings

An embedding of a graph G on a surface Σ is a drawing of G on Σ such that no edges cross. If the surface is then cut along the edges of G, the surface is decomposed into regions called the *faces* of G. An embedding is said to be a *2-cell embedding* if every face is equivalent to an open disc. Groups & Graphs requires that all embeddings constructed are 2-cell embeddings. It can draw graphs embedded on the plane, sphere, projective plane, and torus. It can determine whether a given graph is planar, but currently it does not have the ability to determine whether

an arbitrary graph is projective or toroidal. However, such graphs can be constructed by G&G and drawn. For small graphs, all distinct embeddings on the torus or projective plane can be found by an exhaustive back-tracking algorithm. Effective drawing algorithms for graphs on the sphere and torus have also been implemented. The book by Kocay and Kreher [10] contains a chapter on graph embeddings.

Dual Graph

The *dual* graph of an embedding G, is a graph whose vertices are the faces of G. Adjacent faces of G are adjacent vertices of the dual. Dual graphs can be constructed for planar, spherical, projective, and toroidal embeddings.

Graph Layout

Algorithms for finding a drawing of a graph on the plane, sphere, projective plane, and torus [11] are available.

Truncation

Given an embedding of G on Σ, the truncation of G is formed by replacing each vertex v of G by a cycle, determined by the cyclic order of the edges incident on v. This results in an embedding of a larger graph.

A number of other algorithms are available such as the double cover for projective graphs; converting planar graphs to embeddings on the torus, sphere, or projective plane; drawing a planar graph with an arbitrary face as the outer face; rotating a sphere graph; constructing the medial digraph.

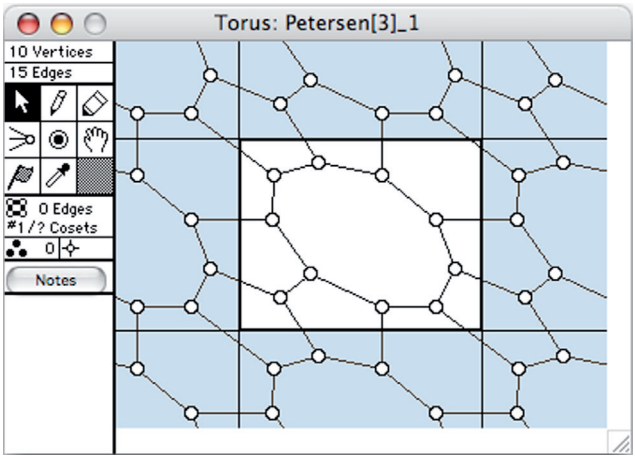

Figure 2, The unique embedding of the Petersen graph on the torus

6. Projective Configurations

An n_3 configuration (see Hilbert and Cohn-Vossen [4]) is a set P of n points and a set L of n lines such that each $p \in P$ is contained in exactly three lines of L, and each $\ell \in L$ contains exactly three points of P. An n_3 configuration can sometimes be drawn in the projective plane, such that each $\ell \in L$ is represented by a straight line, and each $p \in P$ is the intersection of three straight lines. There are unique 7_3 and 8_3 configurations, neither of which can be represented by straight lines. When an n_3 configuration cannot be represented by straight lines, it can always be represented by a drawing in which one line is a circle, and the other lines are straight. G&G can construct drawings for n_3 configurations. It can also find their incidence graphs, duals, and automorphism groups. It can also move points and lines, while simultaneously preserving all incidences. Various conics can also be drawn.

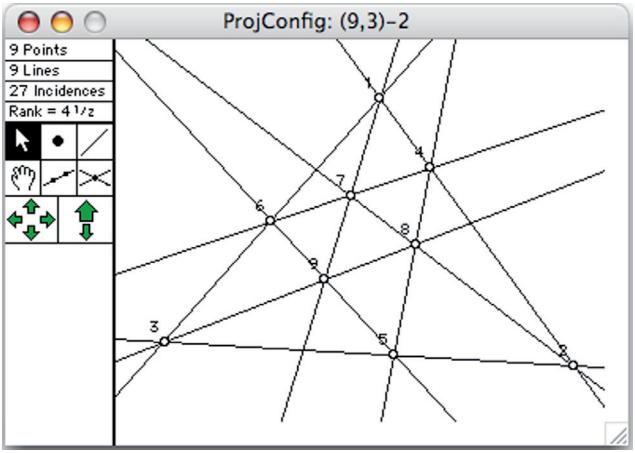

Figure 3, A 9_3 configuration

7. Polyhedra and Fractals

A polyhedron is represented by the planar graph determined by its vertices and edges. It is drawn as a *3D graphics object*. It can be rotated and scaled, and viewed from varying distances and magnifications. The platonic solids are available as predefined polyhedra. The duals and truncations of polyhedra can also be constructed. Polyhedra can also be constructed from graphs drawn on the sphere, or by finding the convex hull of an input file of 3D points. The incremental algorithm (see O'Rourke [12]) is used to find the convex hull.

Some fractals in the complex plane can be constructed. They are useful for studying dynamical systems, or convergence of iterated functions in the complex plane. Fractals are also important in the theory of computability of real numbers. Given a complex function $g(z)$, where z is a complex value (eg., $g(z) = z^2 + c$, where c is a complex constant), the function is iterated:

$$z_{n+1} = g(z_n)$$

starting from some value z_0. If the iterations are unbounded, ie., $|z_n| \rightarrow \infty$, then z_0 is said to be in the escape set of $g(z)$. Otherwise z_0 is said to be in the prisoner set. The *Julia set* of $g(z)$ is the boundary of the prisoner set. The *Mandelbrot set* is the set of all complex values c such that the iterates, starting from $z_0 = 0$, of $g(z) = z^2 + c$ are bounded. See Peitgen, Jürgens, and Saupe [14] for detailed information on fractals. G&G can construct representations of the Mandelbrot set, as well as Julia sets for a number of functions $g(z)$. It can also zoom in to observe finer and finer detail.

References

1. G. Butler, *Fundamental Algorithms for Permutation Groups*, Lecture Notes in Computer Science #559, Springer-Verlag, 1991.

2. H.C. Carr and W. Kocay, "A heuristic for 4-colouring a planar graph", Journal of Combinatorial Mathematics and Combinatorial Computing 46 (2003), 97-112.

3. H.C. Carr and W. Kocay, "An algorithm for drawing a graph symmetrically", Bulletin of the ICA 27 (1999) 19-25.

4. D. Hilbert and S. Cohn-Vossen, *Geometry and the Imagination*, Chelsea Publishing Company, New York, 1983.

5. J. Hopcroft and R. Tarjan, "Efficient planarity testing", J. Assoc. Computing Machinery 21 (1974), 449-568.

6. W. Kocay, "On Writing Isomorphism Programs", book chapter in *Computational and Constructive Design Theory*, Editor: W.D. Wallis, pp. 135-175, Kluwer Academic Publishers, 1996.

7. W. Kocay and P-C. Li, "An algorithm for finding a long path in a graph", Utilitas Mathematica 45 (1994), 169-185.

8. W. Kocay and D. Stone, "An algorithm for balanced flows", J. of Combinatorial Mathematics and Combinatorial Computing 19 (1995), 3-31.
9. W. Kocay, "An extension of the multi-path algorithm for finding hamilton cycles", Discrete Mathematics 101, (1992), pp. 171-188.
10. W. Kocay and D.L. Kreher, *Graphs, Algorithms, and Optimization*, CRC Press, Boca Raton, 2004.
11. W. Kocay, D. Neilson, and R. Szypowski, "Drawing graphs on the torus", Ars Combinatoria 59 (2001), 259-277.
12. J. O'Rourke, *Computational Geometry in C*, Cambridge University Press, 1994.
13. C.H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization*, Dover Publications Inc., Mineola, New York, 1998.
14. H-O. Peitgen, H. Jürgens, D. Saupe, *Chaos and Fractals*, Springer-Verlag, 2003.
15. R.E. Tarjan, Depth-first search and linear graph algorithms, SIAM J. of Computing 1 (1972), 146-160.
16. S.G. Williamson, "Embedding graphs in the plane – algorithmic aspects", Annals of Discrete Mathematics 6 (1980), 349-384.