AUTOMATED VISUAL REPRESENTATION OF LINEAR GRAPHS

R. Brien Maguire
University of Regina

Abstract

Visual representation of graphs is not essential to graph theory. However, the ability to model relationships pictorially has led to the use of graph theoretic techniques in many different applications. While computers normally work with a numeric representation of a graph such as its incidence matrix, drawing a picture from such a representation is a very tedious process.

Previous graph processing systems have incorporated data structures designed to facilitate manipulation of graph theoretic entities. Even systems using a graphic display have simply replaced pen and paper by light pen and display screen. The user must still draw pictures through a slow trial and error process. A graph processing system has been developed which, through a combination of heuristic techniques and semi-automatic procedures, actually generates the visual representations.

The display techniques discussed are based on symmetry in graphs as represented by the automorphism group of a graph. The automorphism mappings indicate resemblance or symmetry between elements of the graph. Interpreting these automorphisms, it is possible to produce pictures of the graph illustrating the same symmetries.

REPRÉSENTATION VISUELLE AUTOMATISÉ DES GRAPHES LINÉAIRES

Résumé

La représentation visuelle des graphes n'est pas essentielle à la théorie des graphes. Toutefois, grâce à la possibilité de visualiser les relations, les techniques de la théorie des graphes sont maintenant appliquées dans de nombreux domaines. Les ordinateurs travaillent normalement avec une représentation numérique d'un graphe comme sa matrice des incidences, mais l'obtention d'une image d'une telle représentation est un processus très complexe.

Les systèmes de traitement des graphes incorporaient jusqu'à maintenant des structures de données facilitant la manipulation des entités de la théorie des graphes. Même dans le cas des systèmes utilisant un affichage graphique, le crayon et le papier étaient simplement remplacés par un photostyle et un écran, de sorte que l'utilisateur effectuait toujours ses tracés par la méthode fastidieuse des essais et des erreurs. On a mis au point un système de traitement des graphes qui produit effectivement des représentations visuelles, à l'aide d'une combinaison de techniques heuristiques et de méthodes semi-automatiques.

Les techniques d'affichage dont on traite sont basées sur la symétrie dans les graphes, telle que représentée par le groupe des automorphismes d'un graphe. Les automorphismes indiquent une ressemblance ou une symétrie entre les éléments du graphe, et il est possible en les interprétant de produire des images du graphe illustrant les mêmes symétries.

AUTOMATED VISUAL REPRESENTATION OF LINEAR GRAPHS

R. BRIEN MAGUIRE
UNIVERSITY OF REGINA

1. INTRODUCTION

A wide variety of application areas employ graph theoretic techniques to model relationships which are represented as pictures. Electrical networks, chemical structures, computational models and communication networks are applications where this approach has been used. One reason is the number of algorithms available for manipulating the graphs. Equally important, however, is that the structure of the graphs can be represented pictorially.

Efficient computer manipulation of graphs requires that they have a numerical data representation. However, incidence or adjacency matrices and other numeric forms of graphs usually have little meaning for humans. Therefore, the engineer or mathematician attempts to translate the numeric output of his programs into pictures, seeking to obtain a deeper insight and understanding of the properties and structure of the results.

Conversion of numeric data into pictures is a tedious trial and error process of drawing and redrawing. The rough picture is modified repeatedly, moving vertices and edges until the result is satisfactory or until one runs out of time, paper or patience. Graph processing systems which allow the user to draw graphs on a graphic display facilitate this process.[5] However, the user must still draw the pictures himself. Such systems only act as an expensive eraser. Methods are needed to use the power of the computer to produce these pictures from the non-pictorial representation of the graph.

We have developed an interactive graphics system, GSYM, and a set of display routines which attempt to do this. GSYM provides the facilities to create, manipulate and display linear graphs. The display routines convert a numeric representation of a graph into a picture. While the process is not fully automatic, user involvement consists primarily of editing pictures already created by the computer.

2. BACKGROUND

The GSYM system and the display routines evolved from an investigation into the representation of a class of cyclically 4-connected cubic planar maps generated by Faulkner.[1] The problem was to devise a method whereby pictures of these maps could be drawn on a display screen without producing each manually.

Faulkner catalosued the sraphs accordins to the number of resions in the sraph. For every resion size n>5, there exists a map consistins of two polysons of n-2 edses each such that each vertex in one of the polysons is adjacent to exactly one vertex in the other polyson. These maps were drawn as concentric polysons. More complex maps were handled by a heuristic routine which searched the sraph for a rins structure consistins of n-sons. If a rins of 4-sons were not found, larser polysons were allowed into the rins. Once a rins was found, any remainins vertices were added to the display in an orderly fashion, buildins towards the centre of the screen. Fisure 1 shows a representation of a 25 resion map obtained in this manner.

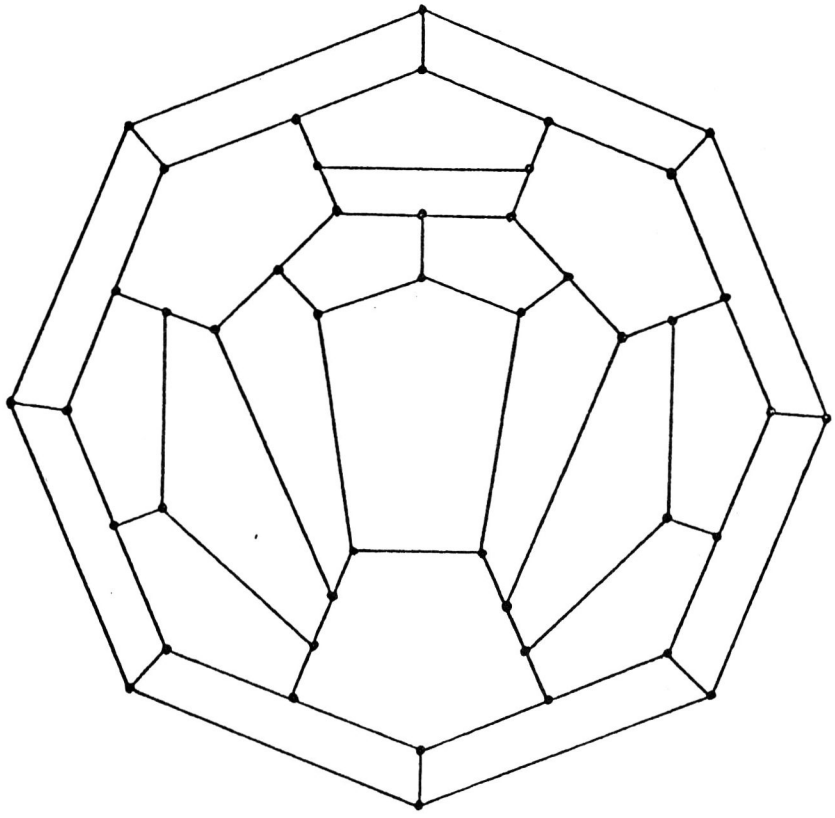

FIGURE 1: 25 REGION RING STRUCTURE

This representation routine had several shortcomins. It misht not find a suitable rins structure. It simply searched for appropriate polysons until a combination worked or until directed to stop, usins larse amounts of computer time. Moreover, and more serious, was that the user could not alter the pictures produced. Finally, the representation was only suited to one class of sraphs. This lack of flexibility led to the development of GSYM and the investisation of alternative representation techniques.

3. GSYM SYSTEM

GSYM, for Graph SYMmetry, is an interactive graphics system which allows a user to draw and manipulate graphs on an IBM 2250 Graphic Display Unit.[4] GSYM was designed as a graphics tool for research into using a computer to generate visual representations of graphs. It does not compare with existing graph processing languages[2,3] as it is not a programming language per se, although it does use macro-like commands for operating on graphs in certain situations. Its primary function is to facilitate the creation and testing of visual representation techniques.

3.1 System Operation

GSYM displays the operations a user may initiate at any given moment and the user selects an option with the light-pen. The system is said to be in the 'reset' state when the option list in Figure 2 is displayed. Once a basic option is selected, all possible suboptions are displayed. For example, Figure 3 shows the suboptions for the ADD operation. The user chooses the element to be added to the graph. Positioning of elements is done with the light-pen.

USE LIGHT-PEN TO SELECT OPTION:

ADD
ALTER
BACKUP
DECLARATION
DELETE
DISPLAY
ENQUIRE
HARD COPY
*** HALT ***
MISCELLANEOUS FUNCTIONS
MODE
MOVE
POINT
ROTATION
TRANSLATION

FIGURE 2: RESET STATE DISPLAY

USE LIGHT-PEN TO SELECT ELEMENT TO ADD:

VERTEX
EDGE
DIRECTION
DIRECTED EDGE
LABEL

*** PFK 30 TO RESET ***

FIGURE 3: ADD OPTION DISPLAY

A similar process is followed for all operations. That is, GSYM awaits a response after displaying the current menu. This format guides the user and eliminates the possibility of system error owing to invalid input. To reduce the frequency of user error most option lists contain a brief instructive note defining the nature of the current operation and the responses allowed. Moreover, an operation may be cancelled in the middle of the command sequence and the reset state reached by simply pushing one of the programmed function keys. Thus, the system is deliberately very forgiving, making it feasible to become proficient in its operation through actual 'hands-on' use. These features are important because as a research tool, GSYM is intended for graph theorists and mathematicians.

Figure 2 illustrates the range of available operations. The basic manipulation options are addition, deletion and alteration of graph entities. The graphs are composed of vertices, edges and arrows. An undirected edge becomes a directed edge by the addition of an arrow(direction). The move, rotation and translation commands manipulate the form of the graph. The user is able to move elements about the screen or relocate the entire graph with the translate option. The rotation command allows the graph to be rotated about any point in the 3-dimensional cube in which it is defined.

3.2 Property Lists

Vertex and edge properties are specified using the DECLARATION option. Each vertex and edge in the graph may have an associated property list. These property lists are used to associate data such as edge weights with individual vertices and edges. The user gives each property a name when declaring it. This name can later be used to reference particular property values. The property lists serve a dual function as they are also used by a list processing subsystem in GSYM to create lists of graph elements. To create a list, a 'head of list' pointer is declared which points to the first element in the list. The vertex or edge property to be used in building the list must be specified at this time. A DECLARATION suboption allows the initial list to be specified. The POINT option allows the user to modify lists. This simple list processing subsystem is adequate for most algorithms.

Since GSYM is a tool for testing representation routines, it must be possible to execute user programs under GSYM. The MISCELLANEOUS FUNCTIONS option displays the available functions. Currently, SYS1-SYS9 invoke system routines such as debug tracing of GSYM, dumping the display buffer, smoothing light-pen edges and inputting graph descriptions. USER1-USER6 are reserved for user programs. User routines are incorporated into the system by replacing

dummy entries in a runtime library. This feature allows the incorporation of representation routines or general graph theoretic algorithms. For example, a program to find Hamiltonian paths could make use of the GSYM list processing subsystem.

3.3 Display Screen Layout

Vertices and edges are displayed as points and lines, their natural representation. An arrow may be positioned anywhere along a directed edge. The positive and negative ends of the edge are specified by arrow orientation, with eight possible orientations corresponding to the eight major points on a compass.

A display screen is limited to two-dimensional pictures. If GSYM were restricted to planar graphs with all representations being planar maps, this would be quite adequate. However, GSYM allows both planar and non-planar graphs. This did not preclude the possibility of generating pictures of non-planar graphs with intersecting edges, but a more serious problem was how to display multiple edges between vertices. With a two-dimensional system it would not be possible to show a graph as a three-dimensional object. For these reasons all GSYM graphs are defined using a three-dimensional coordinate system bounded approximately by the housing of the display screen. Thus an edge is displayed as a line joining the x and y coordinates of the Cartesian (x,y,z) coordinates of its end vertices.

There is one exception to the rule of three-dimensional graph entities, the light-pen edge. The user may draw a freehand representation of an edge using the light-pen. The edge is displayed as short straight line segments. Light-pen edges solve one display problem by allowing multiple edges between a pair of vertices to be drawn as curved lines. However, its real utility is the flexibility in altering pictures. Straight edges can be replaced by curved edges which improve (in the user's opinion) the picture. These edges should only be used in final postediting as they are not three-dimensional entities. As it is not possible to draw a curve in 3-space with the 2250 display, in drawing a light-pen edge the user actually draws the edge as it would be seen on the screen side of the cube in which the graph is defined. Therefore, if a graph with light-pen edges is rotated, all such edges must be redrawn between the new positions of their end vertices.

4. GRAPH SYMMETRY

The most pleasing aspect of the pictures produced by the ring representation was the frequent repetition of symmetry. The possibility of portraying symmetries in graphs suggested using the automorphisms of a graph as a basis for the pictures. The cycles in an automorphism mapping can be interpreted visually as mirror-image and

rotational symmetries. While calculation of the automorphism group of a graph can take large amounts of computer time, various heuristics were implemented to reduce this computation time. The user interacts with the system by dynamically initiating heuristics and varying parameters. This approach made it practical to use automorphisms.

4.1 Representation Routines

Two main representation routines have been developed with GSYM. The first of these is a very general display procedure which serves as a standard against which other routines can be measured. This routine will attempt to draw any graph, independent of any information available about the structure of the graph.

Vertices fixed, that is, mapped onto themselves by an automorphism are displayed as axes of symmetry around which are placed mirror-image vertex pairs, vertices interchanged by the automorphism. Cycles of one or two vertices are thus represented as mirror-image symmetries. Cycles of more than two vertices are treated as rotational symmetries. Each rotation is placed in a different plane. Figure 4 illustrates a simple rotational symmetry. The generality of this approach causes rather cluttered pictures to appear as graphs become more complex. Postediting facilities were added to allow modification of the resultant pictures. The representation routine may be stopped at any point, the picture changed, and the routine allowed to continue. The user controls the placement and relative size of the rotational symmetries. Finally, symmetries may be editted as a unit. For example, moving a vertex in a mirror-image symmetry causes its corresponding vertex to move so as to retain the symmetry. For all its lack of sophistication, this semi-automated process is capable of producing representations of comparable quality to those done manually. Moreover, it does so in a fraction of the time required to draw the same graphs by hand.

One disadvantage of this approach is that the picture quality varies with the automorphism selected. The user soon develops an intuitive feeling for automorphisms that will give a good picture. In general, the best displays are produced from automorphisms with relatively few fixed vertices. While the procedure reveals clearly the symmetries in the graph, it depends on the user to group these symmetries around a pleasing visual framework. The biggest advantage of this approach is its moderate success with many classes of graphs. However, it does not allow for the fact that, in many cases, users will be working on applications where the graphs involved share common properties and structures that could aid in producing pleasing pictures.

In contrast to this approach, a representation routine was developed for trees. Trees are a common class of graphs found in many applications. Structurally, they are well defined in that the vertex set can be partitioned using vertex distance from the centre or bi-centre of the tree. Symmetry information was combined with this fact to produce pictures such as the one in Figure 5. The result was a routine that generates excellent pictures of trees with little or no postediting.

5. CONCLUSIONS

The objective of using a computer to automatically produce pictures of graphs was only partially met. The representation methods used required postediting of the pictures. However, postediting is reduced considerably by incorporating more information about graph structure in the display procedure. This would suggest a parallel to the development of special purpose programming languages, that is, the creation of specialized techniques suitable to specific classes of graphs or applications.

GSYM is being converted to an IMLAC/PDP11 system. Planned improvements include redesign of the list processing system and the addition of data tablet and joystick support for positioning and drawing. While a computational assist will eventually be necessary, this will wait until more work is done on the display procedures. One possibility is to further automate the picture generation by having the computer select the automorphisms or by combining symmetry information from several automorphisms. Picture formatting could be improved by considerations such as balancing the area bounded by polygons.

REFERENCES

1. Faulkner, Gary Bruce. Recursive Generation of Cyclically K-Connected Cubic Planar Graphs. Ph.D. Thesis. Department of Combinatorics and Optimization, University of Waterloo (1971).

2. Friedman, Daniel P., Dickenson, David C., Fraser, John J., and Pratt, Terrence W. GRASPE 1.5: A Graph Processor and its Applications. Department of Computer Science, University of Houston (1969).

3. Hurwitz and Citron. GRAF: Graphic Additions to FORTRAN. Proceedings SJCC 1967, 553-558.

4. IBM System/360 Component Description IBM 2250 Display Unit Model 1. Form A27-2701-2, IBM Data Processing Division, White Plains, N.Y.

5. Wolfbers, Michael S. An Interactive Graph Theory System. Moore School Report No. 69-25, University of Pennsylvania (June 1969).

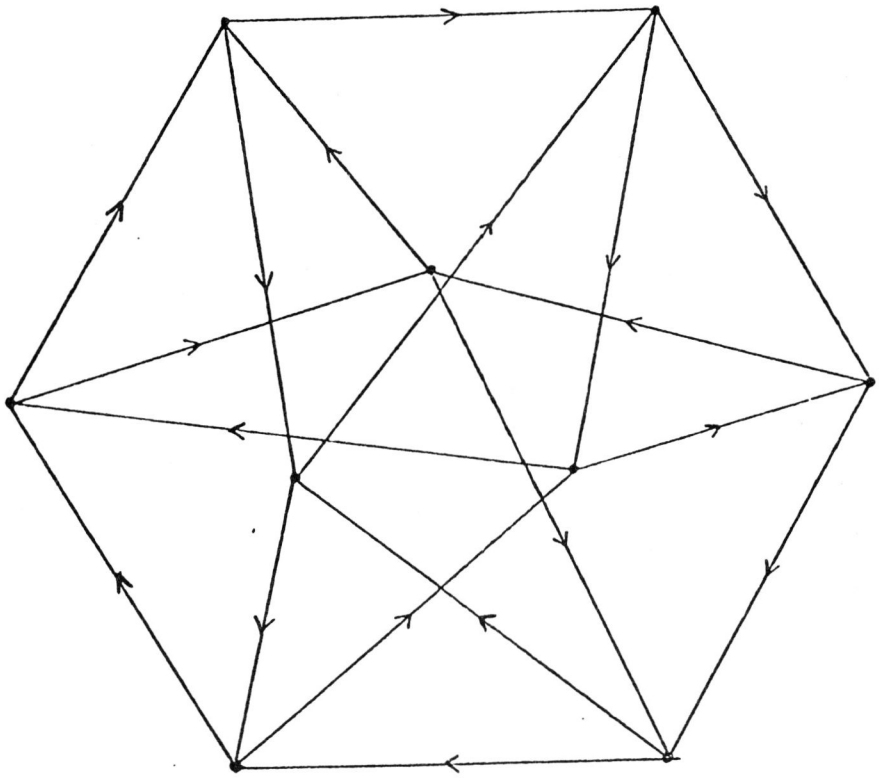

FIGURE 4: ROTATIONAL SYMMETRY

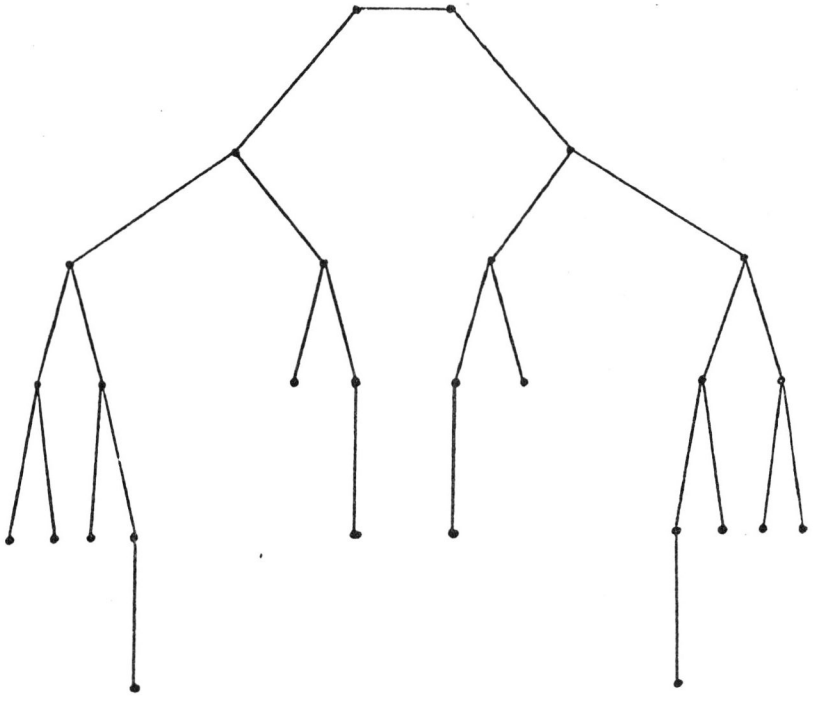

FIGURE 5: TREE WITH SYMMETRY