Software Systems for Research in Graph Theory

JAY BAGGA AND ADRIAN HEINZ
Department of Computer Science
Ball State University
Muncie, Indiana 47306, USA
e-mail: *jbagga@bsu.edu*, *aheinz@bsu.edu*

Abstract

In this paper, we present several graph theory related software systems that we have developed. These systems have been used in learning and research. The systems feature drawing and manipulation of graphs as well as execution of graph algorithms. The systems are *JGraph*, a Java-based system for creating graphs and running graph algorithms; *Colossus*, a visibility graph system; *Manohar*, a system for computing graceful labelings of graphs (with special emphasis on trees) and *Graph Algorithm Constructor*, which allows the creation of graph algorithms by drawing flow diagrams instead of writing source code. We also describe some examples in which the empirical data generated from these systems have allowed us to discover fundamental properties of graphs.

Keywords. Graph algorithms software
2000 Mathematics Subject Classification: 05C

1 Introduction

The large number of applications of graph theory and graph algorithms to a wide range of areas such as computer science, mathematics, engineering, business, geographic information systems, bioinformatics and several others have motivated a rapid growth in the fields of graph theory and graph algorithms during the last four decades. A large number of resources in graph theory such as books, research journals, web sites and other online resources are now available. Graduate and undergraduate programs in computer science, engineering, mathematics and several others regularly include courses on graph theory, graph algorithms and their applications. Students in these courses, researchers in pure and applied graph theory, and professionals who need to experiment with graph algorithms often have varying backgrounds. In particular, students and researchers need to work with graphs and implement graph algorithms but they may not have the necessary programming skills to do so.

The goal of our project is to create software systems for easy manipulation and experimentation with graphs and graph algorithms. Such software systems can be used for learning and teaching, for implementing graph algorithms, for applications, and for conducting research where experimentation with graphs and empirical evidence are needed. These systems should allow students and practitioners to experiment with graphs and construct and implement graph algorithms without programming.

In this paper we describe our recent work and progress of our project. Section 2 gives details of several systems that we have developed and used in our teaching and research activities. In Section 3 we present some applications to illustrate how we have used the systems in our research. We also give a summary of the current status of our project and list future goals and enhancements to the systems.

2 Software Systems

In this section we describe our software systems in detail. While we have found these systems highly useful in our research and teaching, it must be mentioned that the systems are themselves research projects in varying stages of development.

2.1 JGraph

JGraph is a system for creating graphs and running graph algorithms. It has been primarily used for teaching and research in graph theory. The system provides a graphical user interface in which the user can create graphs, modify their structure and execute graph algorithms. Several algorithms have been implemented including planarity testing, planarity drawing, Prüfer encoding, Dijkstra's shortest path, DFS and BFS, and finding blocks of a graph.

The current version (5.0) of the system provides the ability to use GraphML to read and write graphs. GraphML is a comprehensive and easy-to-use file format for graphs. It consists of a language core to describe the structural properties of a graph and a flexible extension mechanism to add application-specific data. See http://graphml.graphdrawing.org/ for more details. The advantage of using GraphML lies in the fact that this format is a standard and therefore many other graph applications use it. This allows JGraph to read files containing graphs created in another graph application which uses GraphML. JGraph has been developed entirely under Java and therefore it can be run on any platform. A screenshot of JGraph is shown in Figure 1.

We refer the reader to [7, 10] for more detailed descriptions of JGraph.

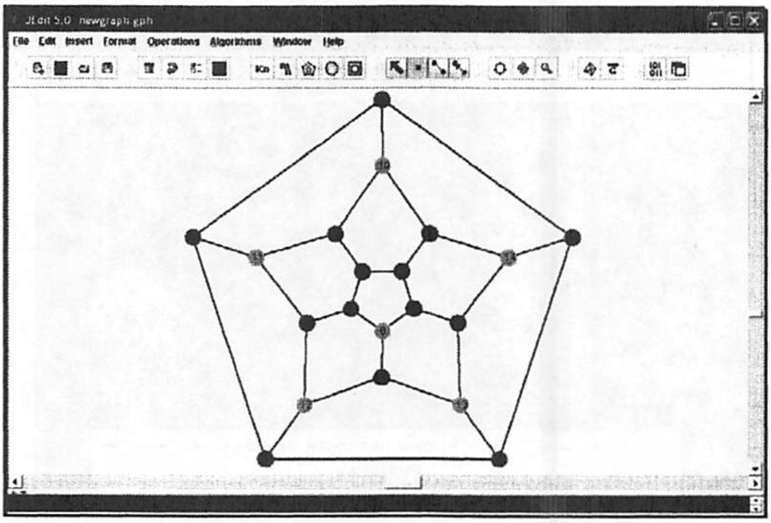
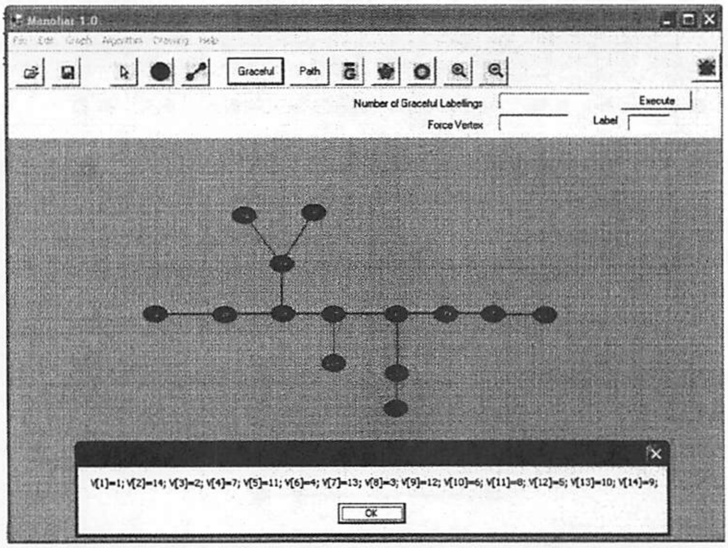


Figure 1: Screenshot of JGraph v5.0.

2.2 Manohar

Manohar is a system for computing graceful labelings of graphs. For the sake of completeness we include the definition of a graceful labeling. For a connected graph G with m edges, we consider a vertex labeling $f : V(G) \to \{0, 1, 2, ..., m\}$ to be an injective map. This induces an edge-labeling of G where an edge xy is given the label $|f(x) - f(y)|$. We say that f is a *graceful labeling* if the induced edge labels are distinct, so that the edges are labeled $1, 2, ..., m$ in some order. A graph G is graceful if it has a graceful labeling. The field of graceful labelings is a very active area of research. Please see [14] for an excellent survey. The well-known Ringel-Kotzig conjecture [14] states that all trees are graceful.

The graphical user interface of Manohar allows the user to easily draw a graph and move vertices and edges around. When the user finishes drawing the graph, it is possible to compute its graceful labeling by clicking on a button. If the graph is graceful, Manohar displays a graceful labeling on the screen. In its current version, Manohar can compute graceful labelings of trees, cycles, and certain other graphs. A sample screenshot of the computation of graceful labeling for a tree of 14 vertices is illustrated in Figure 2.


Manohar 1.0
File Edit Graph Algorithm Drawing Help
Graceful
Path
Number of Graceful Labelings
Execute
Force Vertex
Label
OK

```
V[1]=1; V[2]=14; V[3]=2; V[4]=7; V[5]=11; V[6]=4; V[7]=13; V[8]=3; V[9]=12; V[10]=6; V[11]=8; V[12]=5; V[13]=10; V[14]=9;
```

Figure 2: Screenshot of Manohar

Even though Manohar is able to find graceful labelings of trees and certain graphs, it can only find a limited number of them. Nevertheless, recent improvements to Manohar include the implementation of original algorithms that find all graceful labelings of cycles as well as paths. It is well-known that paths P_n are graceful. Rosa [18] proved that a cycle C_n is graceful if and only if $n \equiv 0 \ mod \ 4$ or $n \equiv 3 \ mod \ 4$. To illustrate how Manohar has helped in our research in graceful labelings, we include below a description of our algorithms and some data generated by those.
Overview of the Algorithm for cycles.

For a cycle C_n (with $n \equiv 0 \ mod \ 4$ or $n \equiv 3 \ mod \ 4$) our algorithm generates edge labels starting at edge label n and proceeding to generate edges labels $n-1, n-2, ...$ and so on down to edge label 1. This is done by assigning appropriate vertex labels. Since the only way to generate edge label n is by labeling two adjacent vertices as 0 and n, our algorithm starts here. The next step is to generate edge label $n-1$. As there are two ways to obtain a positive difference equal to $n-1$, namely $< n-1, 0 >$ and $< n, 1 >$, the algorithm splits the computation into two branches, one with the sub-labeling $< n-1, 0, n >$ and the other one with $< 0, n, 1 >$. The algorithm continues computing edge labels $n-2, n-3, ...$ by splitting into branches until it either finds a graceful labeling for the branch or it reaches a sublabeling for which no graceful labeling is possible. In this last case, the algorithm disregards the computation for that branch and continues computing other branches.

```
11 | 0 11
10 | 10 0 11
9 | 1 10 0 11
8 | 9 1 10 0 11
7 | 2 9 1 10 0 11
6 | 8 2 9 1 10 0 11
5 | 3 8 2 9 1 10 0 11
4 | 7 3 8 2 9 1 10 0 11
3 | 4 7 3 8 2 9 1 10 0 11
2 | 6 4 7 3 8 2 9 1 10 0 11
4 | 3 8 2 9 1 10 0 11 7
3 | 6 3 8 2 9 1 10 0 11 7
2 | 4 6 3 8 2 9 1 10 0 11 7
* 2 | 6 3 8 2 9 1 10 0 11 7 5 [1] > 4
3 | 3 8 2 9 1 10 0 11 7 4
* 2 | 5 3 8 2 9 1 10 0 11 7 4 [2] > 6
2 | 3 8 2 9 1 10 0 11 7 4 6
5 | 8 2 9 1 10 0 11 6
4 | 4 8 2 9 1 10 0 11 6
3 | 7 4 8 2 9 1 10 0 11 6
* 2 | 5 7 4 8 2 9 1 10 0 11 6 [3] > 3
2 | 5 3 -2 7 4 8 2 9 1 10 0 11 6
2 | 3 5 -2 7 4 8 2 9 1 10 0 11 6
3 | 4 8 2 9 1 10 0 11 6 3
2 | 7 5 -2 4 8 2 9 1 10 0 11 6 3
2 | 5 7 -2 4 8 2 9 1 10 0 11 6 3
* 2 | 4 8 2 9 1 10 0 11 6 3 5 [4] > 7
4 | 7 3 -2 8 2 9 1 10 0 11 6
3 | 4 7 3 -2 8 2 9 1 10 0 11 6
2 | 8 2 9 1 10 0 11 6 4 7 3
2 | 4 7 3 5 -2 8 2 9 1 10 0 11 6
3 | 7 3 -2 5 8 2 9 1 10 0 11 6
2 | 7 3 -2 5 8 2 9 1 10 0 11 6 4
* 2 | 7 3 5 8 2 9 1 10 0 11 6 [5] > 4
4 | 3 7 -2 8 2 9 1 10 0 11 6
3 | 8 2 9 1 10 0 11 6 3 7
2 | 8 2 9 1 10 0 11 6 3 7 5
```

Figure 3: Partial output generated by Manohar for a cycle with 11 vertices.

This process finds all graceful labelings of a cycle [12]. Figure 3 displays a sample of the output data obtained by Manohar for C_{11}. The last edge label computed is displayed on the left and the sub-labelings and graceful labelings to the right of | symbol. The graceful labelings are marked with $*$.

We observe that in a graceful labeling of C_n, the vertex labels are n distinct elements of the set $\{0, 1, 2, ..., n\}$ so that exactly one of these numbers is missing from the set of vertex labels. In the output data in Figure 3, this missing number appears to the right of the $>$ symbol.

Table 1 summarizes the number of graceful labelings found for cycles with up to 20 vertices. Each column represents the number n of vertices of the cycle and each row represents the value m of the missing label. This data has been essential in our research to study the structure of graceful labelings. Notice the symmetry in each column. This is due to the fact that the graceful labelings for $m \leq \frac{n}{2}$ are the complements of the graceful labelings for $m > \frac{n}{2}$. It has also been proved [11] that the missing label m lies between $\lceil \frac{n}{4} \rceil$ and $\lfloor \frac{3n}{4} \rfloor$.

		n									
		3	4	7	8	11	12	15	16	19	20
Total #		2	2	12	24	208	492	7,764	20,464	424,784	1,204,540
m	0	0	0	0	0	0	0	0	0	0	0
	1	1	1	0	0	0	0	0	0	0	0
	2	1	0	3	3	0	0	0	0	0	0
	3	0	1	3	6	26	26	0	0	0	0
	4		0	3	6	42	80	299	299	0	0
	5			3	6	36	80	789	1,476	5,932	5,932
	6			0	3	36	120	1,301	3,190	22,210	39,692
	7			0	0	42	80	1,493	3,494	49,714	104,688
	8				0	26	80	1,493	3,646	61,758	162,606
	9					0	26	1,301	3,494	72,778	191,238
	10					0	0	789	3,190	72,778	196,228
	11					0	0	299	1,476	61,758	191,238
	12						0	0	299	49,714	162,606
	13							0	0	22,210	104,688
	14							0	0	5,932	39,692
	15							0	0	0	5,932
	16								0	0	0
	17									0	0
	18									0	0
	19									0	0
	20										0

Table 1: Number of graceful labeling of cycles for $n \leq 20$

Overview of the Algorithm for Paths. The algorithm for the computation of graceful labelings of a path is similar to that used for cycles. The sublabelings are generated as before but these are *linear* rather than *circular*. Also, since there are n vertex labels $0, 1, ..., n-1$ and $n-1$ edge labels $1, 2, ..., n-1$, no vertex label is missing. It was shown in [1] that, for sufficiently large n, a lower bound for the number of graceful labelings of a path is $(\frac{5}{3})^n$. We tested our algorithm for values of n up to 20 and found that the number of graceful labelings for paths is much larger than $(\frac{5}{3})^n$, which suggests that this bound may be improved perhaps by some multiple of n.

We have tested the performance of Manohar using a Dell 4600 computer with 1GB of RAM. Table 2 shows a comparison of our results for cycles with those of Eshghi & Azimi [13]. They used a Pentium IV machine with RAM (256 MB). The performance column displays the time that the application took to find one graceful labeling.

n	Performance (s)	
	Eshghi & Azimi	Our algorithm
8	< 0.01	< 0.01
15	$< .65$	< 0.01
20	< 105.32	< 0.01
55	N/A	< 0.03
72	N/A	< 0.04
112	N/A	< 0.15

Table 2: Performance of our algorithm compared to that of Eshghi and Azimi

The data generated by our algorithms for paths and cycles have led to the discovery of important properties about the structure of graceful labelings [11]. Our next goal is to generalize our graceful labeling algorithm to caterpillars and lobsters and to implement those in Manohar.

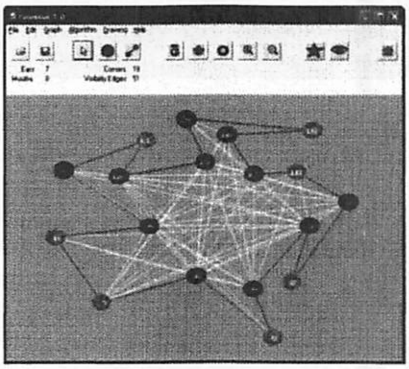
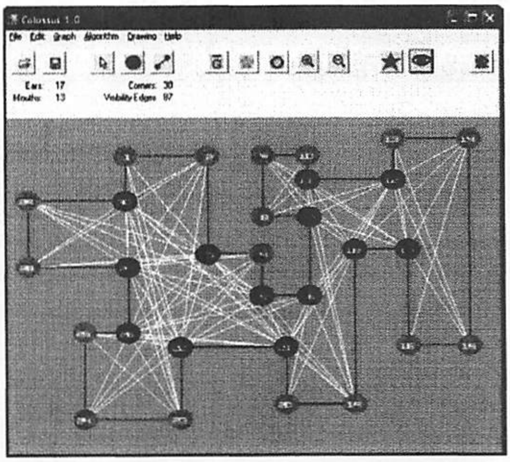


Figure 4: Screenshot of a visibility graph generated by Colossus. Ears and mouths correspond to vertices 2, 5, 7, 8, 11, 15, 18 and 3, 4, 6, 9, 12, 13, 16, 17 respectively

2.3 Colossus

Colossus is a system for determining visibility graphs of polygons or line segments. The study of visibility graphs is an important area of application in computational geometry. We refer the reader to [16, 17] for terminology and basic results. We have done extensive research in this area [2, 3, 4, 5, 6, 8, 9]. Collossus has been quite helpful in our research. A screenshot of Colossus is displayed in Figure 4. The system displays a graphical user interface in which the user can draw a simple polygon. Colossus displays its visibility graph coloring the ears and mouths. See [15] for definitions. The system performs the computation in real-time so that changes to the configuration of the original graph are immediately reflected in its visibility graph.



Ears: 17 Corners: 30
Mouths: 13 Visibility Edges: 97

Figure 5: Visibility graph of an orthogonal polygon.

Another feature of Colossus is its ability to generate visibility graphs of orthogonal polygons. Here, the edges of the polygon are alternatively horizontal and vertical. Orthogonal polygons form an important subclass of general simple closed polygons and are well studied [16]. A sample screenshot is shown in Figure 5. A detailed description of Colossus appears in [10].

3 Summary and Future Work

In this paper we have described several software systems that we have developed for use in research and teaching of graph theory, graph algorithms, and their applications. We described some areas of our research where our systems have been particularly helpful. Students in our graph algorithms classes have used these systems to experiment with graphs and execute graph algorithms. Moreover, these software systems have been essential in our research allowing us to analyze large amount of data to discover important properties of graphs, especially those related to graceful labelings of paths and cycles.

JGraph has been used to teach graph algorithms for more than a decade. The system has been extended over the years by students of several departments including Computer Science, Mathematics and Physics. The reader is invited to visit www.cs.bsu.edu/homepages/gnet for a demonstration version (JEdit 4.2) of this system. See [7] for a detailed discussion of an earlier version of JGraph.

Manohar has been used primarily for research purposes. The system has provided invaluable information in the discovery of important properties of graceful labelings of cycles and paths. Observation of the data has led to the conception of several conjectures and proofs [11].

Collossus has been used for research in visibility graphs. See [2, 3, 4, 5, 6, 8, 9] for details.

Our next goal is to continue enhancing these software systems, with special emphasis on Manohar. The system currently finds all graceful labelings of paths and cycles and we are working on a new version to find all graceful labelings of caterpillars and lobsters.

References

[1] R. E. Aldred, J. Siran and M. Siran, A note on the graceful labelings of path, *Discrete Math.*, **261** (2003), 27-30.

[2] J. Bagga, J. Emert, M. McGrew, and W. Toll, On the Sizes of Some Visibility Graphs, *Congressus Numerantium*, **104** (1994), 25-32.

[3] J. Bagga, L.Gewali, and S. Ntafos, Visibility Edges, Mixed Edges, and Unique Triangulation, *Proceedings of the 13th European Conf. on Comput. Geom.*,(1997), 11.

[4] J. Bagga, S. Dey, L. Gewali, J. Emert, and M. McGrew, Contracted Visibility Graphs of Line Segments, *Proc. 9th Canadian Conf. Comput. Geom.*,1997, 76-81.

[5] J. Bagga, J. Emert, and M. McGrew, Directed Polygon Visibility Graphs, *Congressus Numerantium*, **132** (1998), 61-67.

[6] J.Bagga, J.Emert, and M.McGrew, Directed Polygons as Boundaries of Visibility Graphs, *Congressus Numerantium*, **142** (2000), 57-63.

[7] J. Bagga and A. Heinz, JGraph - A Java based system for drawing graphs and running graph algorithms, *Lecture Notes in Computer Science*, **2265** (2002), Springer Verlag.

[8] J. Bagga, J. Emert, and M. McGrew, Directed Polygons as Boundaries of Visibility Graphs, *Congressus Numerantium*, **142** (2000), 57-63.

[9] J. Bagga, J. Emert, and M. McGrew, Connectivity Properties of Visibility Graphs, *Congressus Numerantium*, **165** (2003), 189-194.

[10] J. Bagga and A. Heinz, Software Systems for Implementing Graph Algorithms for Learning and Research, *ICTACS 2006, Proceedings of the First International Conference on Theories and Applications of Computer Science* (2006), World Scientific Publishers.

[11] J. Bagga, A. Heinz and M. Majumder, Properties of Graceful Labelings of Cycles, *Congressus Numerantium*, **188**(2007), 109-115.

[12] J. Bagga, A. Heinz and M. Majumder, An Algorithm for Graceful Labelings of Cycles, *Congressus Numerantium*, **186**(2007), 57-63.

[13] K. Eshghi and P. Azimi, Applications of mathematical programming in graceful labeling of graphs, *J. Applied Math.*, 1 (2004), 1-8.

[14] Joseph A. Gallian, A Dynamic Survey of Graph Labeling, *The Electronic Journal of Combinatorics* **15** (2008), #DS6.

[15] G.H. Meisters, Polygons have ears, *Amer. Math. Monthly*, **82** (1975), 648-651.

[16] O'Rourke, *Art Gallery Theorems and Algorithms*, Oxford University Press (1987).

[17] J. O'Rourke, *Visibility, Handbook of Discrete and Computational Geometry,* CRC Press (1997).

[18] A. Rosa, On Certain Valuations of the Vertices of a Graph, *Theory of Graphs (Proc. Internat. Symposium*, Rome 1966, Gordon and Breach, N. Y. and Dunod Paris, (1967), 349-355.