# An Interactive Multi-User System for Simultaneous Graph Drawing* *(System Demo)*

Stephen G. Kobourov[1] and Chandan Pitta[2]

[1] Department of Computer Science
University of Arizona
kobourov@cs.arizona.edu
[2] Department of Electrical and Computer Engineering
University of Arizona
chandanp@ece.arizona.edu

**Abstract.** In this paper we consider the problem of simultaneous drawing of two graphs. The goal is to produce aesthetically pleasing drawings for the two graphs by means of a heuristic algorithm and with human assistance. Our implementation uses the DiamondTouch table, a multi-user, touch-sensitive input device, to take advantage of direct physical interaction of several users working collaboratively. The system can be downloaded at `http://dt.cs.arizona.edu` where it is also available as an applet.

## 1 Introduction

Simultaneous drawings of multiple graphs are a useful visualization technique when different relationships are defined on the same set of objects, or when a relationship evolves through time. The objects are represented by graph nodes and the relationships are represented by graph edges. In simultaneous drawings, the placement of the graph nodes is the same in all the drawings, in order to preserve the viewer's mental map. Thus, it is more difficult to obtain good node placement for simultaneous drawings of two or more graphs, compared to the case when only one graph is to be displayed.

Even in the case when only two graphs are given, and individually they are planar, it is not always possible to find consistent node positions that realize plane drawings for each graph. It is not known whether pairs of graphs from a large number of classes allow simultaneous, straight-line, crossing-free embeddings. To aid in the design of algorithms for simultaneous plane drawings for certain classes of graphs and also to help in finding counter-examples (pairs of graphs that cannot be realized) we designed an interactive, multi-user system for manipulating simultaneous drawings of pairs of graphs.

* This work is partially supported by the NSF under grant ACR-0222920 and by ITCDI under grant 003297.

Motivation for this problem comes from applications where it is often necessary to visually compare two relationships. Evolutionary trees on the same set of species are often constructed in computational biology. Biologists spend countless hours pouring over tree drawings to determine the most likely evolutionary branches. The problem is particularly difficult when the drawings of different trees are laid out independent of each other.

### 1.1 Related Work

The problem of simultaneous embedding of planar graphs was introduced in [2], where it is shown that pairs of paths, cycles, and caterpillars can always be realized, while for general planar graphs and even outerplanar graphs this is not always possible. Modified force-directed methods are used to visualize general graphs simultaneously such that the mental map is preserved up in [7]. Conceptually, the problem of simultaneously embedding graphs is the reverse of the geometric thickness problem [5].

The TreeJuxtaposer is a system designed to support the comparison task for large trees [14]. A tool for visualizing large numbers of evolutionary trees on the same set of species is presented in [11].

Traditional informal definitions of aesthetically pleasing graph drawings include features such as straight-line segments for edges, few if any crossings, and display of symmetries. In crossing minimization, the problem is to find a drawing with the minimum number of crossings. The problem is NP-Complete [8] but there has been a great deal of research on heuristic algorithms [9]. Graph planarization [13] is often used together with careful reinsertion of edges.

The Human Guided Search (HuGS) framework described in [1, 10] is an interactive, or human-in-the-loop, optimization system. It leverages people's abilities in areas in which they outperform computers, such as visual and strategic thinking. Users can steer interactive optimization systems towards solutions which satisfy real-world constraints. HuGS has been applied to graph drawing problems in [12]. The DiamondTouch table is introduced in [4] and it has been used for an interactive, multi-player game [3] and for gestural interaction [15].

### 1.2 Our Contributions

We present an interactive multi-user system for simultaneous graph drawing. The system uses the DiamondTouch table, and allows for collaborative work of up to four users. We also provide a heuristic algorithm that attempts to minimize the number of crossings. The algorithms can be used on the entire graphs or on subsets of nodes. The users can stop the algorithm, move nodes around and restart it with the updated positions. Thus, the users can help the algorithm move out of a local minimum, or guide the algorithm towards a more aesthetically appealing solution. Alternatively, if the users get stuck in a local minimum, the algorithm can be started from a random position that may lead to a better solution. Finally, our system works not only with the DiamondTouch

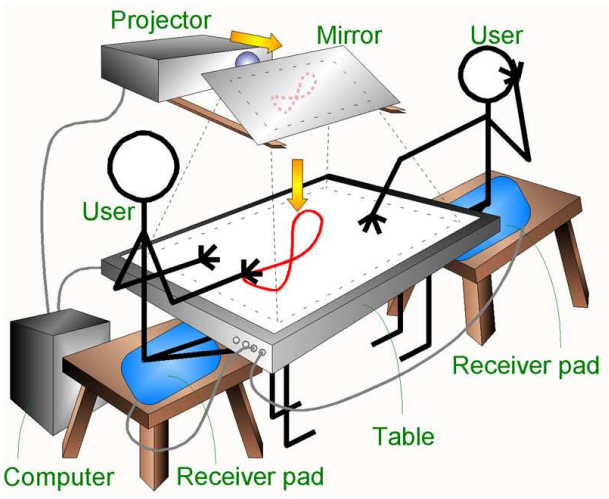


**Fig. 1.** Conceptual DiamondTouch table setup.

table, but also as a Java desktop application, or as a Java applet. The system is operational at `http://dt.cs.arizona.edu`.

## 2 The DiamondTouch Table

The DiamondTouch table [4] from Mitsubishi Electric Research Laboratories (MERL) is a desktop device that allows up to four users to simultaneously manipulate virtual objects. Users can move objects around on the table by touching and dragging them with their fingers. The purpose of the table is to allow several people to interact with a program at the same time and to do so using their hands rather than more common input devices such as mice. The conceptual setup is shown in Fig. 1.

The DiamondTouch table not only detects multiple users, but also identifies which user is touching where on the table. The table is physically large at 32" x 24" and allows several users to work together comfortably; see Fig. 2. Under the surface of the table, there is a grid of antennae. Each antenna transmits a signal to the computer that corresponds to the strength of the capacitance between the user and table. The capacitance is greatest when the user is in direct contact with a particular antenna: a circuit is completed from the antenna, through the user's body, through the receiver pad on which the user is sitting or standing, and back into the table.

The table is designed to be used with an ordinary desktop PC or laptop. It sends the data from the antennae to the DiamondTouch SDK drivers through the USB port, allowing the software to examine the data and to determine where

**Fig. 2.** Physical setup of the DiamondTouch table setup with two users untangling graphs.

on the table the user's fingers are located. The table is not a touch-screen: it has no ability to display output. Instead all images which would normally appear on the display monitor are routed to a video projector which projects them onto the surface of the table with the aid of a mirror and some painstaking calibration.

## 3 Our System

The input to the system consists of two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ defined on the same set of nodes, $V_1 = V_2$, or a subset of a larger common set, $V_1 \subseteq V$ and $V_2 \subseteq V$. The goal is to obtain aesthetically pleasing simultaneous layouts for both graphs.

In the case where $G_1$ and $G_2$ are planar, the goal is to obtain a node configuration that realizes plane drawings for each graph. That is, we are looking for a point set $P$ and bijective function $m : V \rightarrow P$, that maps the set of nodes to

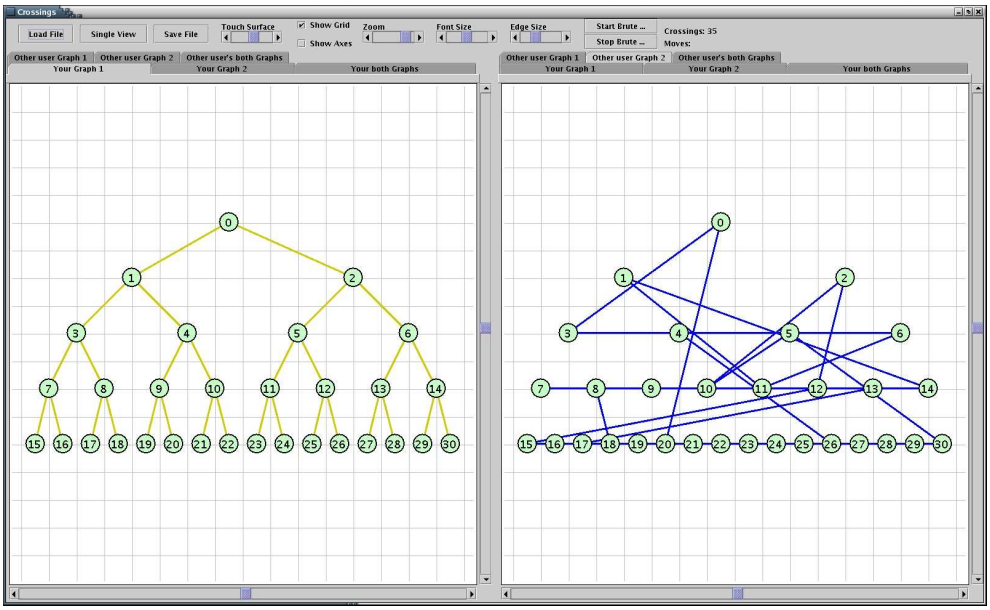


**Fig. 3.** System interface with split view.

points in $\mathcal{R}^2$ such that in a straight-line drawing of $G_1$ on $P$ using the mapping $m$ there are no crossings, and independently in a straight-line drawing of $G_2$ on $P$ using the mapping $m$ there are no crossings; see Fig. 4.

In the case when the two graphs cannot be realized simultaneously as plane drawings, the goal is to obtain symmetric straight-line drawings with as few edge crossings as possible. Note that edge crossings are acceptable if in each pairwise edge crossing one of the edges is from $E_1$ and the other from $E_2$.

The system overview is shown in Fig. 3. The system requires an input file, which contains node and edge information about the two graphs. The graphs are then displayed on the table and users can interact with the system in various ways. Some of the interactions possible are:

- loading and storing graphs via input/output files;
- selecting single-view or split-view;
- selecting drawings to show in the view ($G_1$, $G_2$, or both);
- calling a heuristic crossing-minimization algorithm on both graphs;
- calling the same algorithm on selected parts of the graphs;
- interrupting the algorithm and manually repositioning nodes;
- zooming in and out, or scrolling across larger areas;
- changing colors and sizes of nodes and edges.

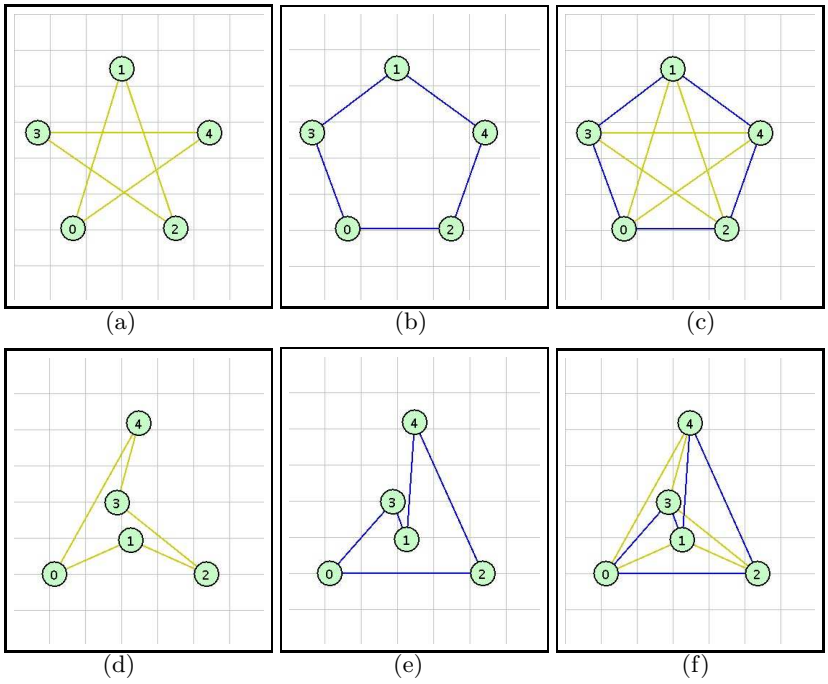


**Fig. 4.** (a-b) Initial drawings of $G_1$ $G_2$ with crossings in $G_1$; (c) Combined view of both drawings; (e-f) Crossing-free drawings of $G_1$ and $G_2$; (f) Combined view of both drawings.

### 3.1 Examples

If the union graph, $G = (V_1 \cup V_2, E_1 \cup E_2)$, of the input pair of graphs, $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ is a planar graph, then our problem has a trivial solution. Since $G$ is planar, there exists a plane drawing of it, and hence for each of $G_1$ and $G_2$ independently. However, if the union graph $G$ is not a planar graph, a solution may or may not exist.

Consider the pair of graphs in Fig. 4(a-b). Both $G_1$ and $G_2$ are simple cycles on 5 nodes. Their union is $K_5$ as seen in Fig. 4(c). However, it is easy to find node locations that realize each of the two graphs with straight-lines and no-crossings; see Fig. 4(d-e). The only crossing in Fig. 4(f) is between edges of different graphs.

While pairs of paths, cycles, and caterpillars are easy to simultaneously draw without crossings and using straight-line edges, this is not the case for all pairs of planar graphs. In fact, it is not known whether two trees can be simultaneously drawn without crossings and using straight-line edges. With this in mind, we experimented with different classes of trees. The pair of trees in Fig. 5 is defined in such a way, that the union of the two graphs contains a subdivision of $K_n$ for any $n$. For the cases when $n \leq 4$ it is fairly straight-forward to obtain by hand straight-line, crossing-free simultaneous drawings. The pen-and-paper solution is

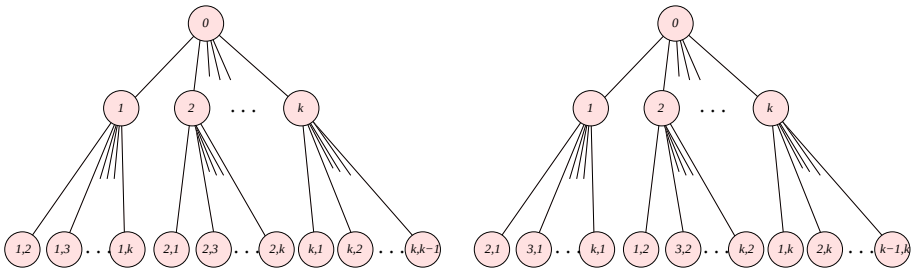


**Fig. 5.** A class of pairs of trees on $n^2 - 2n + 2$ nodes whose union contains a subdivision of $K_n$.

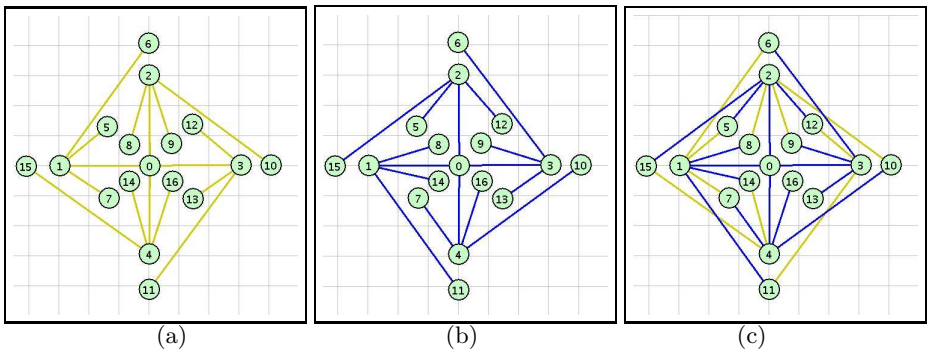


**Fig. 6.** The union of two trees contains a subdivision of $K_5$. (a-b) Crossing-free drawings of $G_1$ and $G_2$; (c) Combined view of both drawings.

difficult to find for $K_5$ and $K_6$. For these two cases our system helped us greatly; see Fig. 6-7.

It is also known that there exist pairs of outerplanar graphs that cannot be realized simultaneously [2]; see Fig. 11. Thus, while it is not possible to design an algorithm for simultaneously realizing pairs of general planar graphs, in many cases solutions do exist. Moreover, no polynomial time algorithms are known for determining whether two planar graphs have a simultaneous embedding or not. Our system can be helpful in gaining insight into the problem and in bridging the gap between the classes of graphs for which algorithms for simultaneous embeddings exist and those for which such embeddings are not possible.

### 3.2 Different Graph Views

Our system offers several different ways to view the input graphs. The main choice in selecting a view is whether it will be a single-view or a split-view. Regardless of the choice, the views can show graph $G_1$, or graph $G_2$, or both

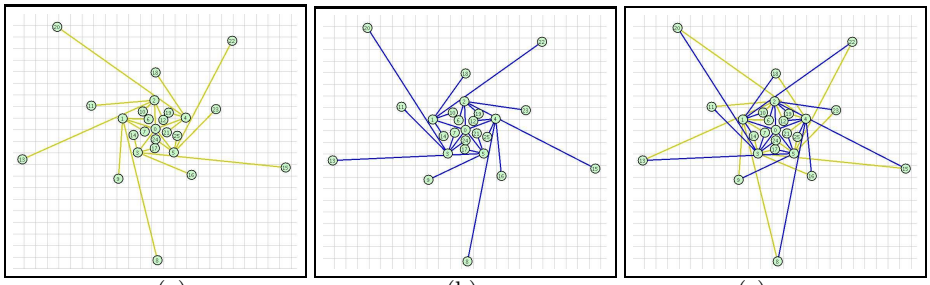


**Fig. 7.** The union of two trees contains a subdivision of $K_6$. (a-b) Crossing-free drawings of $G_1$ and $G_2$; (c) Combined view of both drawings.

graphs at the same time. The split-view with $G_1$ in one and $G_2$ in the other seems the most useful for the purpose of untangling graphs. This view is useful when two groups of people simultaneously work on untangling the two graphs. When a node (and its adjacent edges) is moved in one of the drawings, it also moves in the other drawing. Showing both drawings at the same time allows a user to see the impact of the move in both drawings. A counter keeps track of the current number of crossings. To aid the user in identifying the two graphs, the edges of $G_1$ are colored red and the edges of $G_2$ are colored blue.

### 3.3 Heuristic Crossing Removal

Given a crossing pair of edges from the same graph, $e_1 = (p, q)$ and $e_2 = (r, s)$ we employ a crossing removal strategy consisting of three node-manipulating operations: `flip`, `shrink`, and `rotate` (FSR strategy). We briefly describe the three operations in the FSR strategy below.

The `flip` operation consists of flipping the positions of two nodes that are not endpoints of the same edge. This implies that given crossing pair of edges $e_1 = (p, q)$ and $e_2 = (r, s)$ , there are 4 possible flips. Without loss of generality,

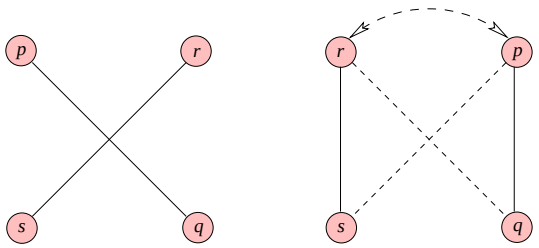


**Fig. 8.** The `flip`$(e_1, e_2)$ operation.

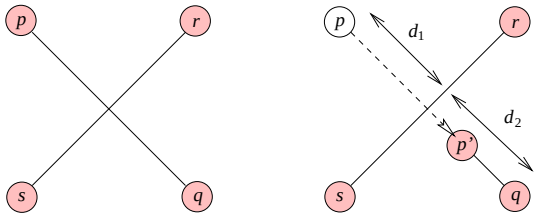


**Fig. 9.** The `shrink`$(e_1, e_2)$ operation.

consider the case where $p$ and $r$ are flipped; see Fig. 8. It is easy to see that flipping the position of two nodes that are not endpoints of the same edge removes the crossing.

The `shrink` operation is performed on edges. It is attempted for each endpoint of each of the edges in the crossing edge pair $e_1 = (p, q)$ and $e_2 = (r, s)$. Without loss of generality, consider the case where the operation is performed on node $p$; see Fig. 9. Let $d_1$ ($d_2$) be the distance from $p$ ($q$) to the intersection point of $e_1$ and $e_2$. The shrink operation for $e_1$ at node $p$ results in moving $p$ along the edge $e_1$ in the direction of $q$ for a of distance $d_1 + k * d_2$ to its new position $p'$, where $k$ is a parameter in the range $0 < k < 1$.

The `rotate` operation is attempted for each node in the crossing edge pair $e_1 = (p, q)$ and $e_2 = (r, s)$. Again, consider the case when the operation is performed on node $p$. Let $\theta$ be the angle determined by the intersection of the lines passing thorough the points $(p, q)$ and $(p, r)$; see Fig. 10. We rotate $p$ around $q$ at an angle $\theta + \epsilon$ to its new position $p'$, where $\theta \leq \epsilon \leq 2\pi$.

Each of the operations in the FSR strategy can be executed a number of times on a particular crossing. Some of them are also parametrized by $k$ and $\epsilon$ for `shrink` and `rotate`, respectively. The three operations are attempted on all

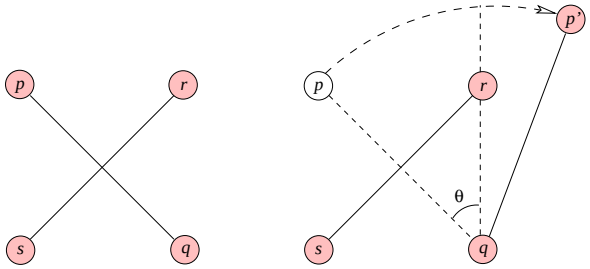


**Fig. 10.** The `rotate`$(e_1, e_2)$ operation.

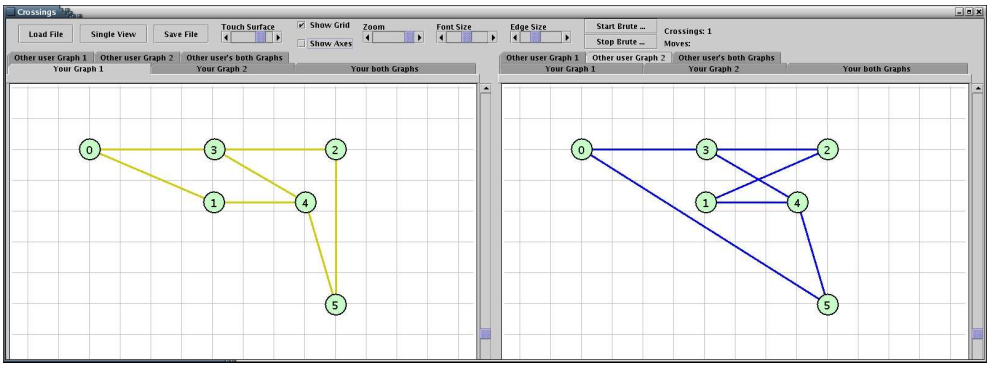


**Fig. 11.** An example of two outerplanar graphs with no simultaneous embedding.

of the undesirable crossings until either they are all removed or we have reached a local minimum.

## 4 Conclusion and Future Work

We have presented an interactive multi-user system for drawing graphs simultaneously. While the system is designed for the DiamondTouch table, it is also available as a Java application, and as a Java applet at `http://dt.cs.arizona.edu`. With the aid of our system we were able to untangle many pairs of graphs that had stumped us in the past. We also used the system successfully, to come up with counter-examples for the cases where simultaneous embedding is not possible. Formal proofs that the pairs of graphs in Fig. 11 and Fig. 12 can be found in [2] and [6], respectively. However, there are many other examples that we have neither been able to realize simultaneously, nor prove that it cannot be done. Some of them are available at the URL above and can be experimented with.

Currently the heuristic algorithm for minimizing the crossings in the simultaneous drawings of the two graphs relies on simple heuristics. We would like to explore better heuristic algorithms, or leverage algorithms and heuristics from traditional crossing-minimization. Finally, we would like to design a brute-force algorithm which can be used to implement a fully functioning HuGS system.

## 5 Acknowledgments

We would like to thank Joe Marks and Kathy Ryall of MERL for supplying us with the DiamondTouch table and for their great help with all the information we needed about it. We would also like to thank Christian Duncan, for coming up with a challenging class of trees (Fig. 5) for our system, and Yuhong Liu, Quanfu Fan, and Joe Schlecht for countless hours of untangling graphs.

## References


1. D. Anderson, E. Anderson, N. Lesh, J. Marks, K. Perlin, D. Ratajczak, and K. Ryall. Human-guided simple search: Combining information visualization and heuristic search. In *Proceedings of the Workshop on New Paradigms in Information Visualization and Manipulation*, pages 21–25, 1999.
2. P. Brass, E. Cenek, C. A. Duncan, A. Efrat, C. Erten, D. Ismailescu, S. G. Kobourov, A. Lubiw, and J. S. B. Mitchell. On simultaneous graph embedding. In *8th Workshop on Algorithms and Data Structures*, pages 243–255, 2003.
3. C. Collberg, S. G. Kobourov, S. Kobes, B. Smith, S. Trush, and G. Yee. Tetratetris: An application of multi-user touch-based human-computer interaction. In 9th International Conference on Human-Computer Interaction (INTERACT), pages 81–88, 2003.
4. P. Dietz and D. Leigh. Diamondtouch: A multi-user touch technology. In *Proceedings of the 14th annual ACM symposium on User interface software and technology*, 2001.
5. M. B. Dillencourt, D. Eppstein, and D. S. Hirschberg. Geometric thickness of complete graphs. *Journal of Graph Algorithms and Applications*, 4(3):5–17, 2000.
6. C. Erten, S. Kobourov, and P. Moravsky. Simultaneous drawing of planar graphs with relaxed constraints. Technical Report TR2004-014, Univeristy of Arizona, 2004.
7. C. Erten, S. G. Kobourov, A. Navabia, and V. Le. Simultaneous graph drawing: Layout algorithms and visualization schemes. In *11th Symposium on Graph Drawing (GD)*, pages 437–449, 2003.
8. M. R. Garey and D. S. Johnson. Crossing number is NP-complete. *SIAM J. Algebraic Discrete Methods*, 4(3):312–316, 1983.
9. C. Gutwenger and P. Mutzel. An experimental study of crossing minimization heuristics. In *Proceedings of the 11th Symposium on Graph Drawing (GD)*, pages 13–24, 2003.
10. G. W. Klau, N. B. Lesh, J. W. Marks, M. Mitzenmacher, and G. T. Schafer. The hugs platform: A toolkit for interactive optimization. In *Advanced Visual Interfaces (AVI)*, 2002.
11. J. Klingner and N. Amenta. Case study: Visualization of evolutionary trees. In *IEEE Symposium on Information Visualization (INFOVIS)*, pages 71–74, 2002.
12. N. Lesh, J. Marks, and M. Patrignani. Interactive partitioning. In *Proceedings of the Symposium on Graph Drawing (GD)*, pages 31–36, 2000.
13. A. Liebers. Planarizing graphs - a survey and annotated bibliography. *Journal of Graph Algorithms and Applications*, 5(1):1–74, 2001.
14. T. Munzner, F. Guimbretiere, S. Tasiran, L. Zhang, and Y. Zhou. Treejuxtaposer: scalable tree comparison using focus+context with guaranteed visibility. *ACM Transactions on Graphics*, 22(3):453–462, 2003.
15. M. Wu and R. Balakrishnan. Multi-finger and whole hand gestural interaction techniques for multi-user tabletop displays. In *ACM UIST Symposium on User Interface Software and Technology*, pages 192–202, 2003.

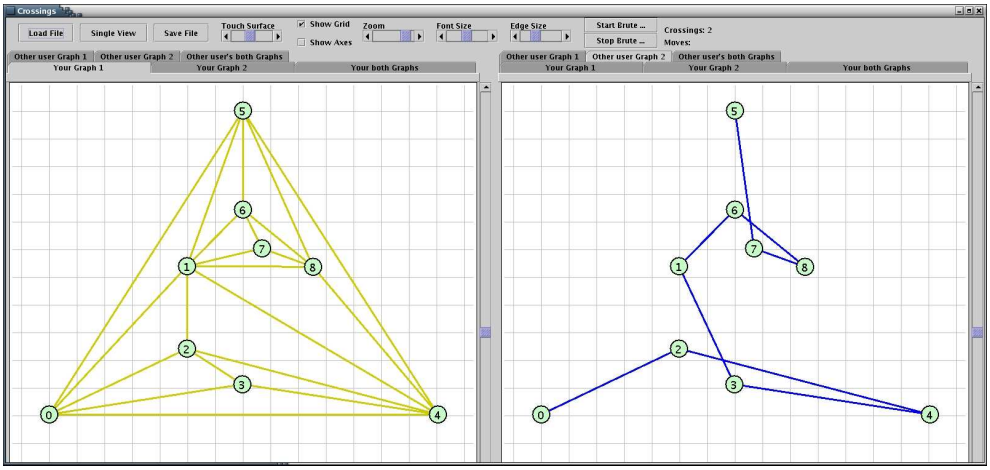


**Fig. 12.** An example of a planar graph and a path with no a simultaneous embedding.

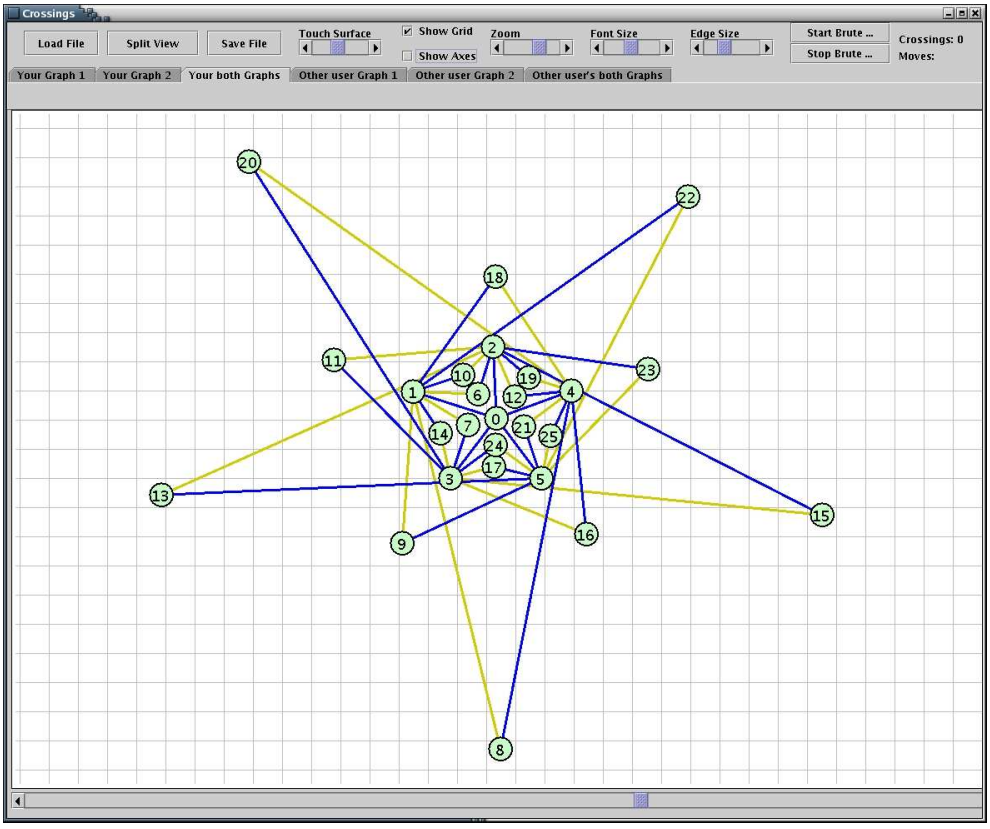


**Fig. 13.** Single view of the two trees in Fig. 7.