Graph Simultaneous Embedding Tool, GraphSET*

Alejandro Estrella-Balderrama, J. Joseph Fowler, and Stephen G. Kobourov

Department of Computer Science, University of Arizona
{aestrell,jfowler,kobourov}@cs.arizona.edu

Abstract. Problems in simultaneous graph drawing involve the layout of several graphs on a shared vertex set. This paper describes a *Graph Simultaneous Embedding Tool*, *GraphSET*, designed to allow the investigation of a wide range of embedding problems. GraphSET can be used in the study of several variants of simultaneous embedding including *simultaneous geometric embedding*, *simultaneous embedding with fixed edges* and *colored simultaneous embedding* with the vertex set partitioned into color classes. The tool has two primary uses: (i) studying theoretical problems in simultaneous graph drawing through the production of examples and counterexamples and (ii) producing layouts of given classes of graphs using built-in implementations of known algorithms. GraphSET along with movies illustrating its utility are available at http://graphset.cs.arizona.edu.

1 Introduction

Drawing multiple graphs simultaneously is a problem motivated by its applications in bioinformatics, social sciences, and software engineering. The large networks defined by multiple relationships make using a single layout impractical. Instead, such networks can be viewed from different perspectives according to the particular structure, behavior, or scale of interest. When looking for common patterns and substructures among the heterogeneous representations of the same data it is essential to preserve the "mental map" of the user. A natural way to accomplish this is to have common vertices and edges laid out in a similar manner throughout the various layouts.

Simultaneous embedding problems are difficult to solve and require extensive manipulation of different instances in order to gain insight. A useful tool is one that allows for the dynamic manipulation of common vertices while accounting for how the edge crossings in each graph can change. In addition, having the ability to visualize each graph separately or as a whole while simultaneously manipulating each graph can allow one to solve complex problems. Finally, having built-in implementations of algorithms related to simultaneous embedding can also aid in further research.

* This work was supported in part by NSF grants CCF-0545743 and ACR-0222920.

I.G. Tollis and M. Patrignani (Eds.): GD 2008, LNCS 5417, pp. 169–180, 2009.

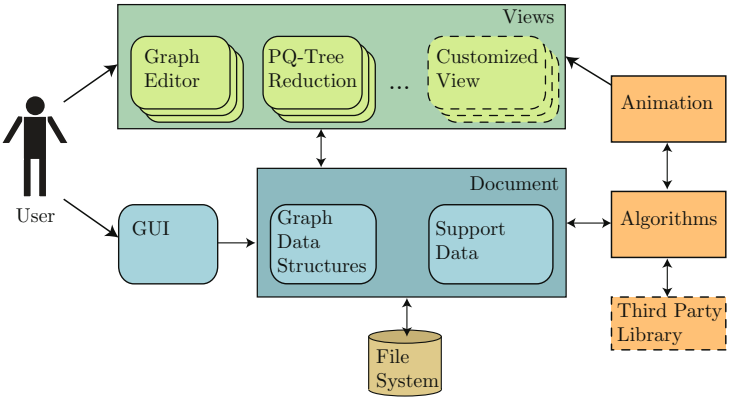

Fig. 1. An overview of the GraphSET system

Our *Graph Simultaneous Embedding Tool*, *GraphSET* meets the above goals, allowing the manipulation of up to eight graphs simultaneously with the capability of displaying each graph separately in its own window. This is an essential feature that has enabled us to solve several simultaneous embedding problems.

A related tool is the *Interactive Multi-User System for Simultaneous Graph Drawing* [15]. It only considers simultaneous geometric embedding of two graphs and the emphasis is on collaboration with the aid of the DiamondTouch device [3]. Another related tool that can be used to obtain simultaneous drawings of graphs using force-directed methods is described in [5].

2 System Architecture

Figure 1 gives a high-level overview of the system architecture of GraphSET. The user can introduce commands using the GUI (menus, dialog boxes, toolbar, etc.) or directly manipulating the view (Graph Editor). When the user makes modifications, they are done in the document (graph data structures, application settings, etc.) and those changes are reflected on every active view of the document. When the modifications are done from the view (such as moving a vertex) the document is modified and reflected back in all active views. The document can be loaded/saved in the file system. Algorithms are called from the document. Some algorithms (such as drawing or recognition) only reflect temporary modifications directly in the view (animation, for example).

Dashed boxes represent plugin components that include customized views (such as a 3D view we have used for studying 3D morphing). The other dashed box corresponds to third-party libraries that can be hooked into the algorithms module via a proxy. For example, we have proxies for LEDA [16] and OGDF (the Open Graph Drawing Framework available at `http://www.ogdf.net`). Algorithms from these libraries are called through these proxies. Overlapping boxes

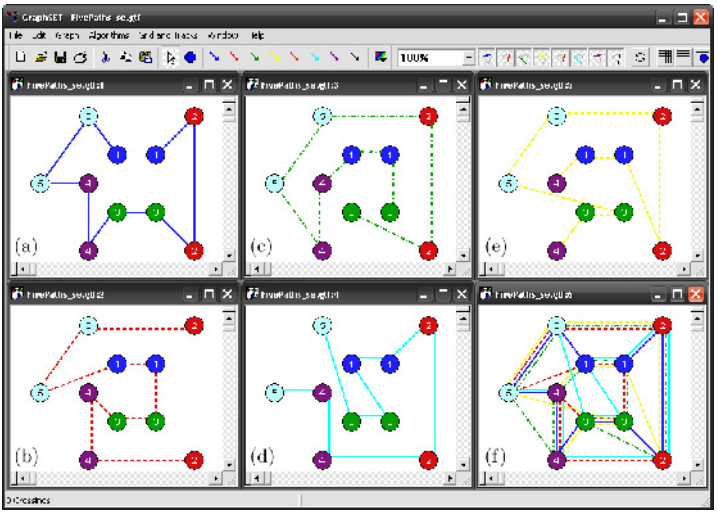

Fig. 2. Example of a simultaneous geometric embedding of five paths: blue path is solid in (a), red is dashed in (b), green is dash-dotted in (c), cyan is light-solid in (d) and yellow is light-dashed in (e). The SGE of all 5 paths is shown in (f).

represent several views of the same type in the document. This allows for the different graph views in which to work with a simultaneous embedding; see Fig. 2. Features like toggling the grid, snapping, and visibility of a given edge set are properties of the graph editor view allowing each to have individual settings.

3 Preliminaries

We begin with a few definitions to clarify the various problems of interest.

Two n-vertex graphs $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ have a *simultaneous embedding with mapping* if, given a bijection $f : V_1 \mapsto V_2$, each graph can be drawn in the plane $\mathbb{R}^2$ without crossings such that for all $v \in V_1$ and $f(v) \in V_2$, v and $f(v)$ are represented by the same point in their respective drawings. If f is not given, but this can be done for some bijection, then G_1 and G_2 are *simultaneously embeddable without mapping*. Unless indicated otherwise, a simultaneous embedding (SE) refers to one with mapping.

A *simultaneous geometric embedding* (SGE) consists of a simultaneous embedding in which only straight-line edges are used. *Simultaneous embedding with fixed edges* (SEFE) is less restricted since edges are drawn with simple curves and common, or *fixed edges*, use the same curve. Clearly, SGE $\subset$ SEFE $\subset$ SE.

The problem of *colored simultaneous embedding* (CSE) is a generalization of simultaneous embedding with mapping in which each V_i is strictly partitioned into k *colors* with respect to a k-coloring of a pointset P. Each vertex of a given color can be mapped to a point of the same color. When $k = n$ this is equivalent

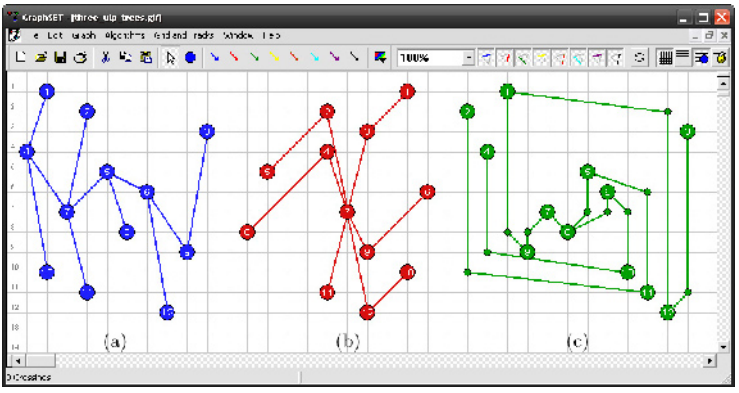

Fig. 3. Layouts of ULP trees: (a) caterpillar, (b) radius-2 star, and (c) degree-3 spider

to simultaneous embedding with mapping, and when $k = 1$, to simultaneous embedding without mapping. Figure 2 is an example of five 5-colored paths on ten vertices in which there are $5 \cdot 2^5 = 160$ possible mappings, one of which is shown.

Finding simultaneous embeddings with paths drawn monotonically uses a restricted form of planarity, called *level planarity*. Only one of the Cartesian coordinates is allowed to change when attempting to find a crossings-free drawing.

An undirected *level graph* $G(V, E, \phi)$ has a *labeling* $\phi : V \mapsto [1..k]$ assigning each vertex to one of k levels so that $\phi(u) \neq \phi(v)$ for every edge (u, v). This prevents any pair of adjacent vertices from being in the same level. In a *level drawing* all the vertices of the same level share the same y-coordinate, placed along a horizontal *track*, and each edge is drawn strictly y-monotone. If G can still be drawn planarly, then G is *level planar*, otherwise, G is *level non-planar*. Any level planar drawing with bends has one without bends [4]. Hence, adding edge bends does not affect the level planarity of a graph.

If G is level planar over all possible labelings, then G is *unlabeled level planar* (ULP). In [6], ULP trees were characterized as consisting of three classes of trees: (i) *caterpillar*s (the removal of vertices that have degree-1 yields a path or an empty graph); (ii) *radius*-2 *star*s (any number of paths of length one or two that all share a common endpoint), and (iii) *degree*-3 *spider*s (three paths that share a common endpoint); see Fig. 3.

4 Applications

In this section, we describe several successful uses of GraphSET. First, we discuss how GraphSET has been used in working with ULP trees [6] and a related problem on colored trees. Second, we consider a pair of trees whose union is homeomorphic to complete graph K_n for $n > 3$ for which there is a pair without a SGE [10]. Third, we discuss how GraphSET has aided in verifying gadgets of

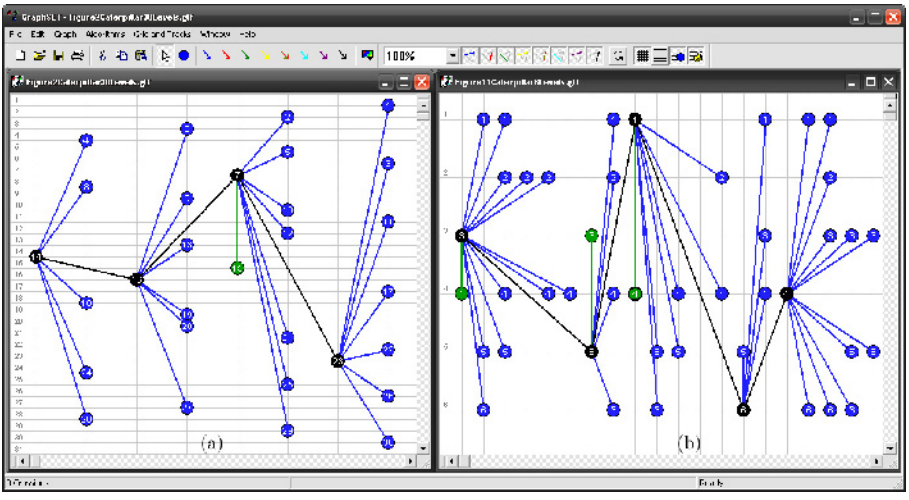

Fig. 4. Layouts of (a) a 25-level caterpillar and (b) a 6-level caterpillar

reductions used to show that deciding whether a graph pair has a SGE is NP-hard and whether a graph triple has a SEFE is NP-complete [7,9]. Finally, we show how GraphSET can be used to find CSE counterexamples, as in [2].

4.1 Unlabeled Level Planar Trees

When there are more vertices than levels, caterpillars are the only class of trees that remains ULP. This allows GraphSET to draw any caterpillar without crossings; see Fig. 4. When there is exactly one vertex per level, GraphSET can also provide level planar layouts of the other two classes of ULP trees; see Fig. 5.

GraphSET also implements the ULP recognition algorithms that highlight the ULP trees by their class. If the graph is not ULP, a subgraph homeomorphic to one of the forbidden ULP trees is highlighted as the user modifies the graph; see Fig. 6. GraphSET has been instrumental in determining correct and implementable algorithms for these purposes. Movies of the tool demonstrating all the ULP tree algorithms can be found at `http://ulp.cs.arizona.edu`.

4.2 Colored Level Planar Trees

Our tool has the feature of allowing the user to snap and lock vertices to tracks in order to investigate not only unlabeled level planar graphs but the planarity of multiple level graphs being simultaneously embedded. Tracks can be colored so that only vertices of that color can be snapped to that track.

As an example of this utility, we consider the open problem of whether a 3-colored tree-path pair always has a SGE. One approach is to attempt to layout the path monotonically. Here each colored track has one vertex of its color.

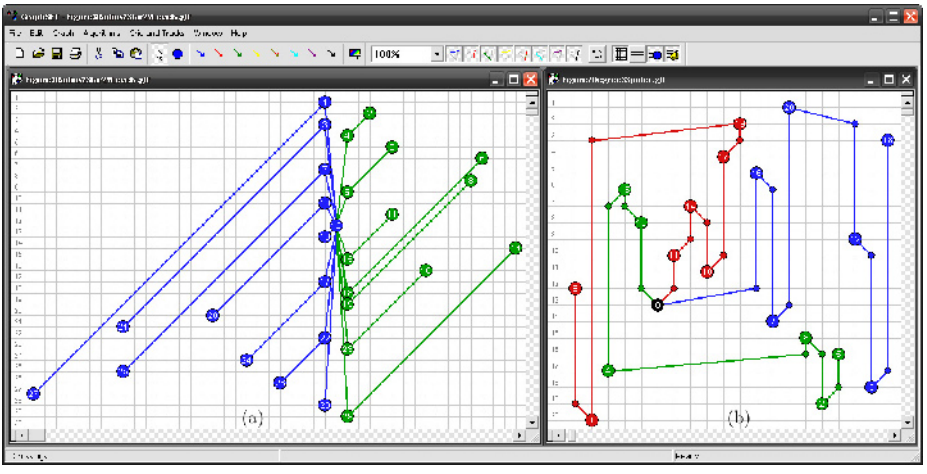

Fig. 5. Layouts of (a) a 30-level radius-2 star and (b) a 20-level degree-3 spider

The idea is to find an algorithm to swap vertices between tracks of the same color until the 3-colored tree becomes level planar. This is not always possible if the tracks are colored sequentially as in Fig. 7(a). However, if the tracks are colored randomly, then it may be possible to find a sequence of swaps in going from a level non-planar assignment as in Fig. 7(b) to a level planar one as in Fig. 7(c). Even in the worst case of sequentially colored tracks there may be relatively few interchanges of colored tracks needed so that a CSE then becomes possible. This would then correspond to paths consisting of relatively few monotonic segments that may have a SGE.

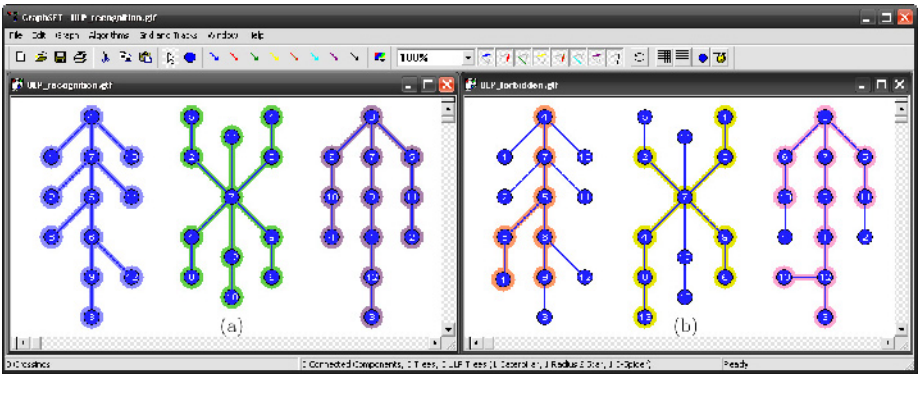

Fig. 6. ULP recognition algorithms highlighting a caterpillar, a radius-2 star, and a degree-3 spider (a) and the forbidden trees T_7, T_8, and T_9 (b)

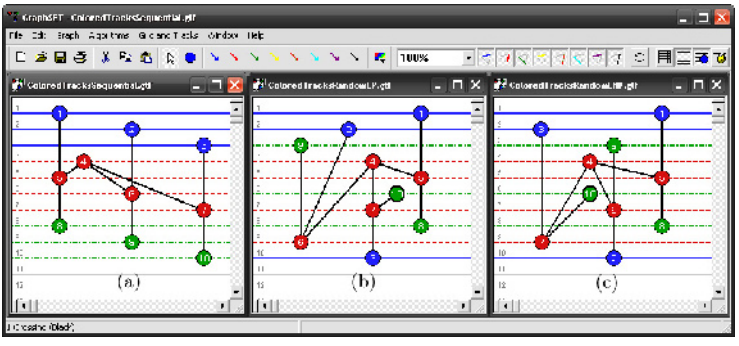

Fig. 7. This 3-colored tree (vertices 1-3 are blue, 4-7 red and 8-10 green) is level non-planar for sequentially colored tracks (tracks 1-3 are blue and solid, tracks 4-7 are red and dashed, tracks 8-10 are green and dash-dotted) as in (a), but may or may not be level planar for randomly colored tracks as in (b) and (c), respectively

4.3 Simultaneous Geometric Embedding of Pairs of Trees

In this section, we consider the simultaneous geometric embedding of two trees $T_1(V, E_1)$ and $T_2(V, E_2)$ on $n^2 - 2n + 2$ vertices whose union contains a subgraph homeomorphic to the complete graph K_n on n vertices for a given $n > 3$. Both T_1 and T_2 have a root vertex labeled '0' that is adjacent to the remaining $n-1$ vertices of V labeled '1', '2', ..., '$n-1$'. In each tree, these $n-1$ vertices have $n-2$ leaves so that each non-leaf vertex has degree $n-1$. Leaves are labeled i, j for $i, j \in [1..n-1]$ and $i \neq j$. In T_1 the vertex labeled $i \in [1..n-1]$ has leaves

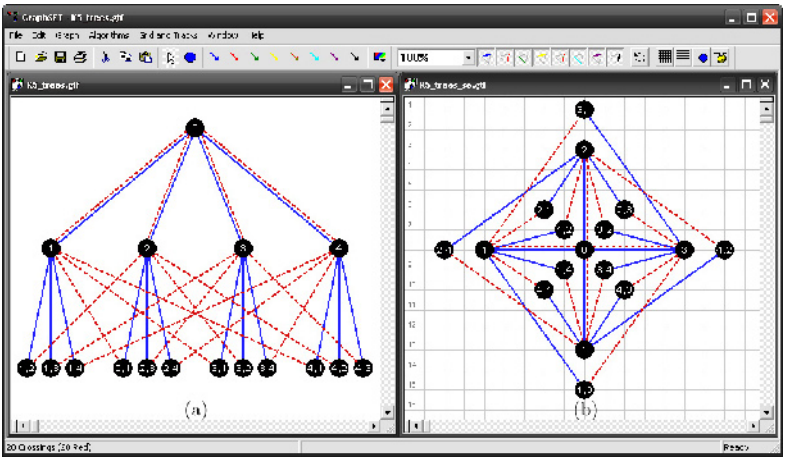

Fig. 8. A pair of trees whose union is homeomorphic to K_5 (a) in which one tree has red edges (dashed) and the other has blue edges (solid) with a SGE shown in (b)

labeled i, j for $j \in [1..n-1]$ such that $i \neq j$. Similarly, in T_2 the vertex labeled $j \in [1..n-1]$ has leaves labeled i, j for $i \in [1..n-1]$ such that $i \neq j$.

The tool is especially useful in this case given that the user can have different windows for each graph. GraphSET maintains the crossing count within each graph while ignoring the crossings of edges from different graphs. Figure 8 shows two trees for the case of $n = 5$ on 17 vertices that illustrates a schema to generate a layout that works up to $n = 6$. When $n > 6$, we found that the root vertex labeled '0' could no longer be centrally located, but rather had to be on the convex hull of the simultaneous embedding. For large values of n these tree pairs do not have a simultaneous geometric embedding, as shown by Geyer *et al.* [10]. It is unknown what is the smallest value of n that forces a crossing; for example, the case $n = 8$ is open.

4.4 Gadgets for Planar 3-SAT Reductions

GraphSET supports multiple edges with different colors. These edges may include bends and can be treated as a single edge (for fixed edges) or as different edges (for multi-graphs). An application of this is the manipulation of gadgets for Planar 3-SAT reductions.

In [9] Gassner *et al.* proved that SEFE is NP-Complete for three graphs. The proof is a reduction using clause gadgets and literal gadgets; see Fig. 9(a). There are two possible embeddings for each literal gadget and these embeddings correspond to true or false values in the matching literals. The argument is that a drawing of the clause without crossings is only possible if one of the literals is true. In the drawing this implies that we can only get rid of a crossing by flipping a literal gadget (changing the embedding of the gadget).

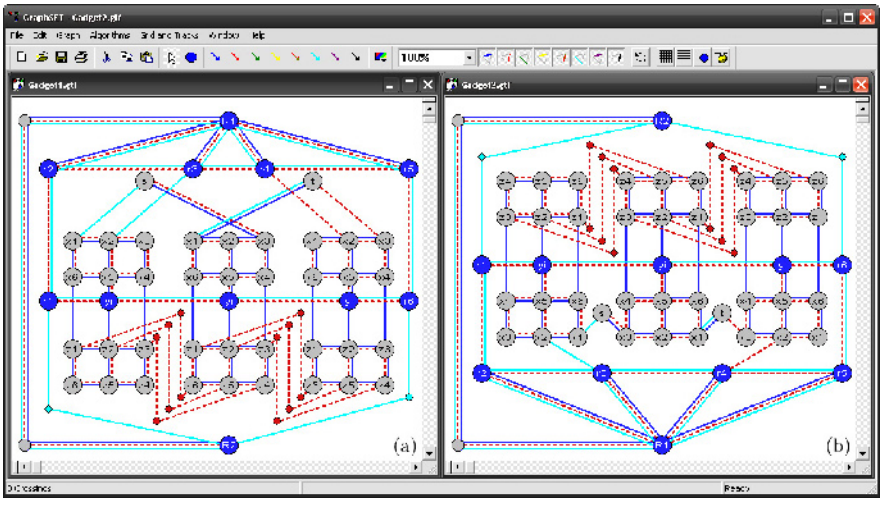

Fig. 9. Gadget for a clause with 3 literals (a) and a SEFE of the gadget in (b)

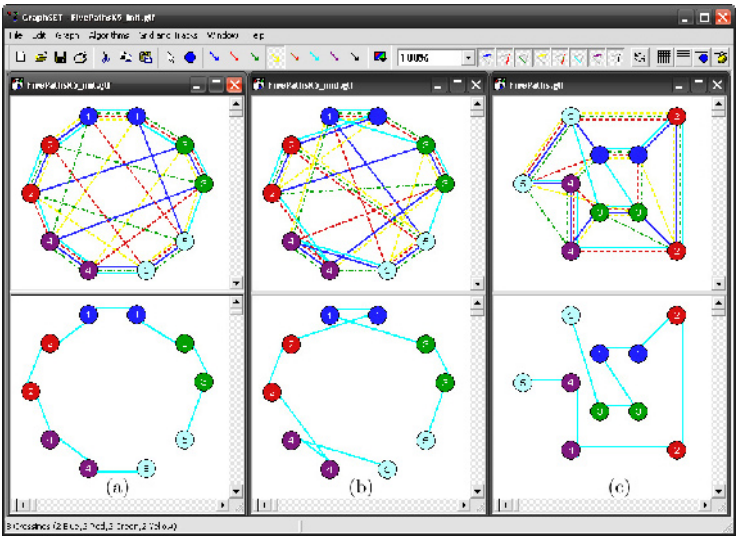

Fig. 10. Five colored paths (blue path is solid, red is dashed, green is dash-dotted, cyan is light-solid and yellow is light-dashed) without a SEFE in (a) and after some swaps among vertices of the same color in (b) have a SGE in (c). The split window shows the cyan path for each step.

GraphSET is useful in exhibiting problems in the gadget construction by finding initially less than obvious embeddings that may break the argument; see for example Fig. 9(b). With the aid of GraphSET the correct reduction was found [7]. The flipping/rotation and the cut/paste operations included in the tool are essential in constructing and manipulating these kinds of gadgets.

4.5 Colored Simultaneous Embeddings

GraphSET was used to build a counterexample of five 5-colored paths on five distinctly colored vertices without a SGE to show that there does not exist a universal pointset for 5-colored paths [2]. One open CSE problem is whether there exists four paths on four colors that do not always have a SGE. We illustrate the difficulty of this problem with a potential alternate counterexample of five 5-colored paths not using distinctly colored vertices with Figs. 2 and 10. Here the five 5-colored paths are on ten vertices in which each path has two vertices of the same color corresponding to its endpoints. As given in Fig. 10(a) a crossing will always occur regardless of the placement of vertices. This is due to the fact that each pair of vertices with the same color is connected by four edges of the other colors. This means that when each of these vertex pairs are contracted they form the example of five paths on five colors in [2].

However, vertices of the same color can exchange adjacencies. The tool lets one swap the adjacency lists between two vertices of the same color in one

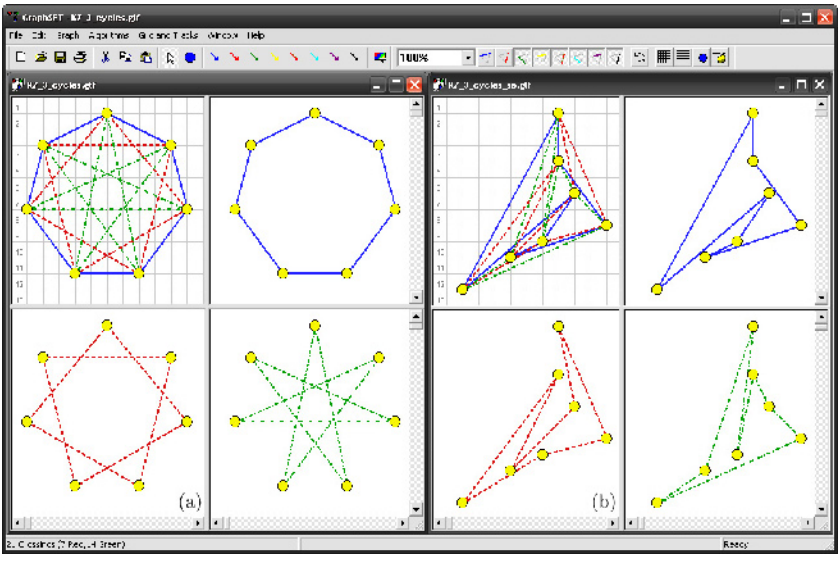

Fig. 11. Three cycles whose union forms a K_7 with no common edges in (a) and the corresponding SGE in (b)

of the graphs. A series of such swaps in the five graphs results in Fig. 10(b), which has the SGE in Fig. 10(c). While this is not the counterexample we are after, it illustrates the utility of GraphSET when attempting to construct such counterexamples.

Another open CSE problem in which GraphSET is very useful is shown in Fig. 11(a). One starts with an arrangement of three cycles whose union forms a K_7. In general, any odd prime p has a decomposition into $(p-1)/2$ cycles whose union forms a K_p in which each edge in the union is in exactly one cycle. This is of interest because the three 6-colored cycles whose union forms a $K_{3,3}$ without a SGE given in [2] are constructed so that each edge in the union belongs to two of the three paths. This forces one of the cycles to have a self crossing.

It is an open problem to find a set of cycles without any common edges that do not have a SGE. While this example for K_7 has a SGE shown in Fig. 11(b), this requires several small angles between pairs of incident edges along the same cycle. We conjecture for sufficiently large p that such a SGE no longer exists.

5 Implementation

GraphSET is a stand-alone Windows application written in C++ that can be downloaded from `http://graphset.cs.arizona.edu`, where the source code is also available. GraphSET can also run under Linux and MacOS using `wine`.

GraphSET contains other related algorithms for graph drawing as support for the previous applications. This includes implementation of the PQ-tree data

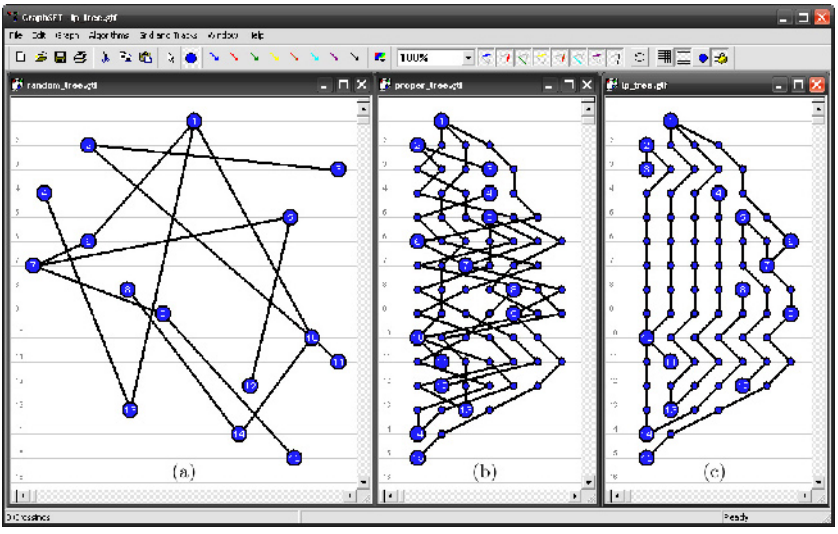

Fig. 12. A random labeled tree (a) made proper (b) with a level planar embedding (c)

structure and the planarity testing algorithm by Booth and Lueker [1] . The level planarity testing and embedding algorithms by Healy *et al.* [11,12] are also available. These algorithms require the graph to be proper, i.e., there are only edges between vertices in consecutive levels. When the graph is non-proper, GraphSET adds dummy vertices along edges; see Fig. 12. The runtime for these algorithms is $O(|V|^2)$ provided the graph is proper.

6 Conclusions and Future Work

We presented GraphSET, a tool that has been valuable in studying problems related to simultaneous embedding. We hope that other researchers interested in these problems will find this tool useful.

While currently GraphSET only includes the recognition and drawing algorithms for ULP trees, we plan to incorporate algorithms for all ULP graphs. We foresee using this tool in the research of minimal level non-planar (MLNP) patterns; the first step is to implement MLNP patterns recognition algorithms for trees [8]. We also plan to incorporate the faster $O(|V| \log |V|)$ level planarity testing and embedding algorithms by Jünger, Leipert and Mutzel [13,14].

Acknowledgments

We would like to thank Martin Harrigan for the explanations on the embedding part of the vertex-exchange algorithm, Markus Geyer in helping finding the 3-colored tree counterexample, and Michael Jünger and Sebastian Leipert for providing us with their level planarity testing code.

References

1. Booth, K., Lueker, G.: Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. J. Comput. Syst. Sci. 13, 335–379 (1976)
2. Brandes, U., Erten, C., Fowler, J.J., Frati, F., Geyer, M., Gutwenger, C., Hong, S., Kaufmann, M., Kobourov, S.G., Liotta, G., Mutzel, P., Symvonis, A.: Colored simultaneous geometric embeddings. In: Lin, G. (ed.) COCOON 2007. LNCS, vol. 4598, pp. 254–263. Springer, Heidelberg (2007)
3. Dietz, P., Leigh, D.: Diamondtouch: a multi-user touch technology. In: 14th ACM Symposium on User interface software and technology, pp. 219–226 (2001)
4. Eades, P., Feng, Q.-W., Lin, X., Nagamochi, H.: Straight-line drawing algorithms for hierarchical graphs and clustered graphs. Algorithmica 44(1), 1–32 (2006)
5. Erten, C., Kobourov, S.G., Navabia, A., Le, V.: Simultaneous graph drawing: Layout algorithms and visualization schemes. In: Liotta, G. (ed.) GD 2003. LNCS, vol. 2912, pp. 437–449. Springer, Heidelberg (2004)
6. Estrella-Balderrama, A., Fowler, J.J., Kobourov, S.G.: Characterization of unlabeled level planar trees. In: Kaufmann, M., Wagner, D. (eds.) GD 2006. LNCS, vol. 4372, pp. 367–369. Springer, Heidelberg (2007)
7. Estrella-Balderrama, A., Gassner, E., Jünger, M., Percan, M., Schaefer, M., Schulz, M.: Simultaneous geometric graph embeddings. In: Hong, S.-H., Nishizeki, T., Quan, W. (eds.) GD 2007. LNCS, vol. 4875, pp. 280–290. Springer, Heidelberg (2008)
8. Fowler, J.J., Kobourov, S.G.: Minimum level nonplanar patterns for trees. In: Hong, S.-H., Nishizeki, T., Quan, W. (eds.) GD 2007. LNCS, vol. 4875, pp. 69–75. Springer, Heidelberg (2008)
9. Gassner, E., Jünger, M., Percan, M., Schaefer, M., Schulz, M.: Simultaneous graph embeddings with fixed edges. In: Fomin, F.V. (ed.) WG 2006. LNCS, vol. 4271, pp. 325–335. Springer, Heidelberg (2006)
10. Geyer, M., Kaufmann, M., Vrtǒ, I.: Two trees which are self-intersecting when drawn simultaneously. In: Healy, P., Nikolov, N.S. (eds.) GD 2005. LNCS, vol. 3843, pp. 201–210. Springer, Heidelberg (2006)
11. Healy, P., Harrigan, M.: Practical level planarity testing and layout with embedding constraints. In: Hong, S.-H., Nishizeki, T., Quan, W. (eds.) GD 2007. LNCS, vol. 4875, pp. 62–68. Springer, Heidelberg (2008)
12. Healy, P., Kuusik, A.: The vertex-exchange graph: A new concept for multi-level crossing minimisation. In: Kratochvíl, J. (ed.) GD 1999. LNCS, vol. 1731, pp. 205–216. Springer, Heidelberg (1999)
13. Jünger, M., Leipert, S.: Level planar embedding in linear time. In: Kratochvíl, J. (ed.) GD 1999. LNCS, vol. 1731, pp. 72–81. Springer, Heidelberg (1999)
14. Jünger, M., Leipert, S., Mutzel, P.: Level planarity testing in linear time. In: Whitesides, S.H. (ed.) GD 1998. LNCS, vol. 1547, pp. 224–237. Springer, Heidelberg (1999)
15. Kobourov, S.G., Pitta, C.: An interactive multi-user system for simultaneous graph drawing. In: Pach, J. (ed.) GD 2004. LNCS, vol. 3383, pp. 492–501. Springer, Heidelberg (2005)
16. Mehlhorn, K., Näher, S.: The LEDA Platform of Combinatorial and Geometric Computing. Cambridge University Press, Cambridge (1999)