Proper and Planar Drawings of Graphs on Three Layers

Matthew Suderman

School of Computer Science, McGill University
msuder@cs.mcgill.ca

Abstract. A graph is *proper k-layer planar*, for an integer $k \geq 0$, if it admits a planar drawing in which the vertices are drawn on k horizontal lines called layers and each edge is drawn as a straight-line segment between end-vertices on adjacent layers. In this paper, we point out errors in an algorithm of Fößmeier and Kaufmann (CIAC, 1997) for recognizing proper 3-layer planar graphs, and then present a new characterization of this set of graphs that is partially based on their algorithm. Using the characterization, we then derive corresponding linear-time algorithms for recognizing and drawing proper 3-layer planar graphs. On the basis of our results, we predict that the approach of Fößmeier and Kaufmann will not easily generalize for drawings on four or more layers and suggest another possible approach along with some of the reasons why it may be more successful.

Layered graph drawings [16] have applications in visualization [2, 9], DNA mapping [17], and VLSI layout [10]. In a layered drawing, the vertices are drawn on a set of horizontal lines called layers, and edges are drawn as straight-line segments between their end-vertices. Depending on the purpose of the drawing, it may also satisfy additional constraints. Common constraints include bounds on the number of layers in the drawing, and restrictions on the edges that may intersect one another.

In this paper, we consider layered drawings that are proper and planar; that is, we consider drawings in which the end-vertices of each edge lie on adjacent layers, and edges intersect only at common end-vertices. Heath and Rosenberg [8] show that the problem of recognizing graphs with proper and planar drawings on layers is $\mathcal{NP}$-complete. In a more restricted version of this problem, the input is not only a graph but also a number $k \geq 0$, and the problem asks whether or not the graph has a proper and planar drawing on k layers. Though the $\mathcal{NP}$-completeness of the original problem implies that this problem is also $\mathcal{NP}$-complete, Dujmović *et al.* [4] show that it can be solved in polynomial time when k is bounded by a constant. Unfortunately, the constants in the running time are impractically large even for $k = 3$.

The difficulty of this problem seems to increase as the number of layers increases. Consequently, this motivates a study of proper and planar drawings on a very small number of layers in hopes of obtaining insights for drawings on a

P. Healy and N.S. Nikolov (Eds.): GD 2005, LNCS 3843, pp. 434–445, 2005.

larger number of layers. Interestingly, this approach has had some limited success for planar layered drawings that are not proper. In particular, Cornelsen, Schank and Wagner [3] show that a graph G has a planar layered drawing on three layers if and only if a certain transformation of G has a drawing on two layers. Using this result, they obtain a linear-time recognition and graph drawing algorithm for three layers.

Proper planar drawings on up to three layers have also been studied. For one and two layers, the drawings are quite simple and it is easy to determine in linear-time whether or not a graph admits such a drawing [5, 7, 15]. For three layers, Fößmeier and Kaufmann [6] also claim to have a linear-time recognition algorithm; however, we will show that, even though their algorithm seems plausible, it contains significant errors.

Following a few preliminary definitions and results in Section 1, we will briefly describe their algorithm in Section 2, and then discuss its flaws in Section 3. We will then describe a new characterization of graphs that have proper and planar drawings on three layers, and derive a corresponding linear-time recognition and drawing algorithm in Section 4.

The overall purpose of our work is not to correct an error, but rather to obtain efficient algorithms for layered graph drawing. Therefore, we would like know if the approach of Fößmeier and Kaufmann [6] can be generalized to obtain recognition and drawing algorithms for four or more layers. The simplicity of their algorithm seems to suggest a positive answer; however, based on the complexity of our attempt to correctly handle all cases, such a generalization would probably be very long and tedious. In Section 5, then, we will conclude by suggesting another approach that may lead to efficient algorithms for layered graph drawing.

1 Preliminaries

In this paper, each graph $G = (V, E)$ is simple, undirected and connected. A graph $G = (V, E)$ is *bipartite* if its vertices can be partitioned into two disjoint sets V_0 and V_1 such that each edge in E has one end-vertex in V_0 and the other in V_1. We call V_0 and V_1 the *bipartition classes* of G and write $G = (V_0, V_1; E)$.

A *leaf* vertex in a graph is a vertex with exactly one neighbor. Any graph that can be transformed into a path $v_1, v_2, \ldots, v_p$ by removing all of its leaves is called a *caterpillar*, and the path $v_1, v_2, \ldots, v_p$ is called the *spine* of the caterpillar. The 2-*claw* is the smallest tree that is not a caterpillar. It consists of a vertex called the *root* that has three neighbors, and each neighbor is additionally adjacent to a leaf. See Figure 1 for a drawing of a caterpillar and a 2-claw.

A *cut-vertex* in a graph is a vertex whose removal disconnects the graph.

A *planar drawing* of a graph is a two-dimensional drawing in which each pair of edges may intersect only at a common end-vertex. A *planar embedding* of a graph defines a clockwise order of the neighbors of each vertex that corresponds to a planar drawing of the graph. A graph is *k-layer planar* if it has a planar drawing in which each edge is drawn as a straight-line between its end-vertices

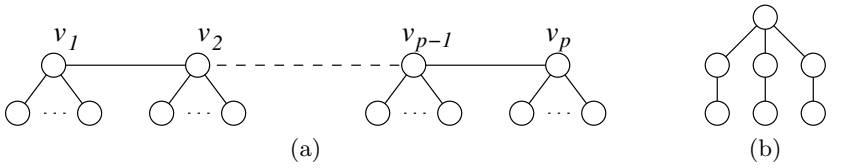


Fig. 1. (a) Caterpillar, (b) 2-Claw

and the vertices lie on k horizontal lines called *layers*. Such a drawing is called a *k-layer planar drawing*. A graph is said to be *proper k-layer planar* if it has a k-layer planar drawing in which the end-vertices of each edge lie on adjacent layers. In other words, the edges in such drawings intersect layers only at points that coincide with the drawings of their end-vertices. Such a drawing is called a *proper k-layer planar drawing*.

As mentioned in the introduction, there is a simple characterization of proper 2-layer planar graphs:

Lemma 1 ([5, 7, 15]). *Let G be a graph. The following are equivalent:*

1. *G is proper 2-layer planar;*
2. *G is a forest of caterpillars;*
3. *G is acyclic and contains no 2-claw; and*
4. *The graph obtained from G by deleting all leaves is a forest and contains no vertex with degree three or greater.*

In this paper, we will prove a similar though much more complicated characterization theorem for proper 3-layer planar graphs.

2 A Recursive Approach for Recognizing Proper 3-Layer Planar Graphs

Rather than present the entire algorithm of Fößmeier and Kaufmann [6], we will describe only the basic approach of the algorithm and then, in the following section, describe in detail only those steps that contain significant flaws.

The algorithm depends on a few simplifying assumptions. First of all, the algorithm assumes that the input graph is bipartite. This is because every proper 3-layer planar graph is bipartite (the vertices on the top and bottom layers in a proper 3-layer planar drawing belong to one bipartition class and the remaining vertices belong to the other), and it is easy to test whether or not a graph is bipartite. The algorithm also assumes that the vertices of some given bipartition class must be drawn on the top and bottom layers. Thus, to recognize all proper 3-layer planar graphs, the algorithm would need to be applied two times, each time with a different bipartition class selected as the one that must be drawn on the top and bottom layers. In the algorithm, we will denote the bipartition classes as V_0 and V_1 and assume that the vertices of V_0 must be drawn on the top and bottom layers.

Given these assumptions, the algorithm divides the recognition problem into several cases and then handles nearly each case recursively. For example, in one case, the input graph G contains a cut-vertex $v \in V_1$ with at least four non-leaf neighbors. If v has more than four non-leaf neighbors, then it is not too difficult to see that G does not have a proper 3-layer planar drawing with v on the middle layer; therefore, the algorithm returns false. Likewise, if v has exactly four non-leaf neighbors and $G - v$ contains three connected components that are each not caterpillars (i.e. not proper 2-layer planar), then G is also not proper 3-layer planar so the algorithm returns false. If G and v pass these two tests, then v has exactly four non-leaf neighbors and $G - v$ contains at most two non-caterpillar components. The algorithm then returns true if and only if each non-caterpillar component (plus v) has a proper 3-layer planar drawing in which v has the smallest or largest x-coordinate of any vertex in the drawing.

We observe that the previous case is handled recursively but with an additional constraint on the position of v. In fact, the input to the algorithm consists not only of a graph but also of a set of vertices called *borders*. This set may contain up to two vertices and the algorithm returns true if and only if the graph has a proper 3-layer planar drawing with one of the vertices in *borders*, if $|borders| > 0$, has the smallest x-coordinate of any vertex in the drawing, and the other vertex in *borders*, if $|borders| > 1$, has the largest x-coordinate of any vertex in the drawing.

Even without describing the remainder of the algorithm, one can see that this basic approach appears to be promising and seems to suggest a very simple algorithm for recognizing proper 3-layer planar graphs. Unfortunately, as we will show in the next section, the algorithm contains significant flaws in the way it handles some of the other cases.

3 Shortcomings of the Algorithm

In this section, we describe some of the errors in the algorithm of Fößmeier and Kaufmann [6] and provide examples of graphs that it either incorrectly accepts or rejects as a result of these errors.

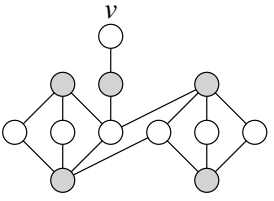
The algorithm divides the problem into four main cases numbered (a)-(d), and each main case is divided into two or more subcases numbered 1, 2, and so on. The cases are handled in alpha-numeric order; thus, for example, the algorithm handles cases (b)-(d) only if case (a) does not apply to the input. The algorithm is called `test` and, as described in the previous section, the input to the algorithm is a bipartite graph $G = (V_0, V_1; E)$ and a set of vertices *borders*.

– Case (a2) states:

> **if** there is a small vertex[1] $v \in V_1$ **then**
> **if** $v \in$ *borders* **then** insert v's neighbor into *borders* **fi**;
> call test($G \setminus \{v\}$, *borders* $\setminus \{v\}$);

[1] A small vertex is a leaf.

Consider the following graph $G = (V_0, V_1; E)$ where the vertices of V_0 are darkened:



The algorithm would first remove v from the graph as described in (a2). The remaining graph $G - v$ is clearly proper 3-layer planar so the recursive call to the algorithm should return true. However, by our characterization theorem in Section 4, this graph is not proper 3-layer planar because the main biconnected component is not safe (see Definition 4).

– Case (c2) does not contain an error, however, we mention it because it is never fully described anywhere in the literature. This case assumes that V_1 contains no vertices with four non-leaf neighbors, contains no vertices that have three non-leaf neighbors and are cut-vertices, but does contain at least one vertex with three non-leaf neighbors. Case (c2) states:

> we need a more special case analysis involving separator edges[2]which is omitted;

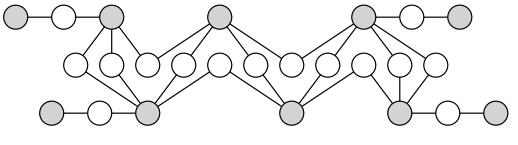
Unfortunately, by definition, the separator edge referred to belongs to a cycle. Based on the fact that the bulk of the complexity in our characterization theorem in Section 4 relates to cycles, we would be very surprised if there is a straightforward way to handle this case.

– Case (d3) states (where G is biconnected and the vertices of V_1 have degree equal to 2):

> let L be the graph obtained by replacing all $v \in V_1$ by edges between its neighbors. **if** L is a ladder graph (an outerplanar graph with completely nested shortcut edges[3]) **then** return true **else** return false;

We assume that the authors mean that, not only must L be a ladder graph, but it must have a drawing in which one vertex of *border*, if $|border| > 0$, has the smallest x-coordinate in the drawing and the other vertex, if $|border| > 1$, has the largest x-coordinate in the drawing.

If this is the case, then the algorithm would reject the following graph even though it is proper 3-layer planar:

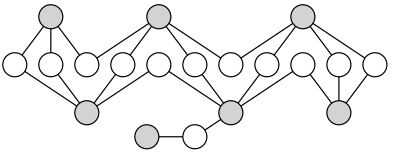


[2] A separator edge is an edge (u, v) such that removing u and v from the graph disconnects the graph.

[3] A shortcut edge in a biconnected outerplanar graph does not lie on the external face.

Since the graph contains four separator vertices in V_0, the algorithm would return false because any attempt to recursively draw the main biconnected component in the graph would require the parameter *borders* to contain all four separator vertices. As mentioned above, *borders* may contain at most two vertices.

If this is not the case (i.e. the algorithm ignores *borders* in case (d3)), then the algorithm would return true for the following graph even though it is not proper 3-layer planar:



We address this difficulty in our characterization by defining safe biconnected components (see Definition 4).

4 Characterizing Proper, 3-Layer Planarity

Our characterization of proper 3-layer planar graphs is based on many of the observations contained in the algorithm of Fößmeier and Kaufmann [6]; however, as we mentioned earlier, correct handling of all cases can be very tedious.

Our characterization consists of constraints on vertices and biconnected components. For the restricted case where the input graph G is a tree, our characterization is very similar to their algorithm: roughly it says that G is 3-layer planar if and only if, for each vertex v in G, at most two connected components of $G - v$ are not proper 2-layer planar. The reason is that, if there are three components that are not proper 2-layer planar, then each component in the drawing of G occupies all three layers. Consequently, one component must be drawn between the other two components. The problem, however, arises when we want to add v to the drawing. Since it is adjacent to all three components, its edge incident on the leftmost component or its edge incident on the rightmost component will cross an edge of the middle component.

In the more general setting, G may not be a tree because even-length cycles are proper 3-layer planar. Consequently, to characterize proper 3-layer planar graphs, we must handle biconnected components containing more than two vertices. As we will show, we can handle biconnected components by generalizing the way we handle vertices in trees. For example, it is not difficult to see that if C is a biconnected component in a proper 3-layer planar graph G, then $G - C$ contains at most two connected components that are not proper 2-layer planar. Unfortunately, this in itself is not sufficient because not all biconnected components are proper 3-layer planar. For an example of one, see Figure 2. Consequently, our characterization must contain additional constraints for biconnected components.

In the following, we describe the constraints on vertices and biconnected components more formally and completely. A vertex or biconnected component that

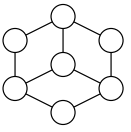


Fig. 2. A biconnected graph that is not 3-layer planar

satisfies these constraints will be called *safe*. We recall that proper 3-layer planar graphs are bipartite so our definitions will apply to bipartite graphs and be given with respect to a given bipartition class of the graph.

Definition 1. *Let V_0 be a bipartition class of a bipartite graph G. A vertex v in G is* safe *with respect to V_0 if:*

1. *$v \in V_0$ and $G - v$ contains at most two components that are not caterpillars (e.g see Figure 3(b)); or*
2. *$v \notin V_0$ and v has at most four non-leaf neighbors, and:*
   (a) *$G - v$ contains at most two components H_1 and H_2 such that $G(V(H_1) + v)$ and $G(V(H_2) + v)$ are not caterpillars.*
   (b) *if v has four non-leaf neighbors or v belongs to a cycle, then $G - v$ contains at most two components H_1 and H_2 such that $G(V(H_1) + v)$ plus a leaf attached to v and $G(V(H_2) + v)$ plus a leaf attached to v are not caterpillars (e.g. see Figure 3(c)).*

It is not too difficult to see that if a bipartite graph $G = (V_0, V_1; E)$ contains a vertex that is not safe with respect to V_0, then G does not admit a proper 3-layer planar drawing in which the vertices of V_0 lie on the top and bottom layers.

A vertex v may be safe but only just "barely safe" because the connected components H_1 and H_2 of $G - v$ mentioned above are in fact not caterpillars. As a result, in every proper 3-layer planar drawing of G, one of these components must be drawn to the left of v and the other component must be drawn to the

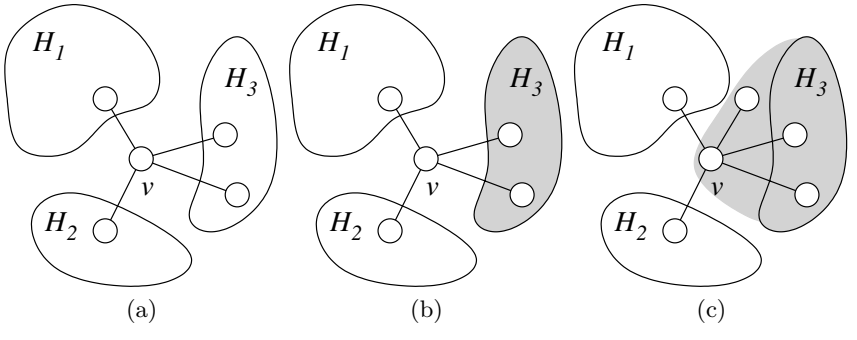


Fig. 3. (a) Suppose that, for some vertex v in $G = (V_0, V_1; E)$, $G - v$ has three connected components H_1, H_2 and H_3. (b) If H_3 is a caterpillar and $v \in V_0$, then v is safe with respect to V_0. (c) If $v \notin V_0$ and $G(V(H_3) + v)$ plus a leaf attached to v is a caterpillar, then v is safe with respect to V_0.

right. Thus, we call v a *connecting vertex* because it "connects" the left part of the drawing with the right part of the drawing.

Definition 2. *A vertex v is a* connecting vertex *with respect to V_0 if:*

1. *$v \notin V_0$ and v has four non-leaf neighbors in G; or*
2. *v does not belong to a cycle and $G - v$ contains two components H_1 and H_2 such that $G(V(H_1) + v)$ and $G(V(H_2) + v)$ are not caterpillars; or*
3. *v belongs to a cycle and $G - v$ contains two components H_1 and H_2 such that the graph containing $G(V(H_1) + v)$ plus a leaf attached to v and the graph containing $G(V(H_2) + v)$ plus a leaf attached to v are not caterpillars.*

To describe the properties of a *safe* biconnected component, it is necessary to know how the biconnected component is connected to the remainder of the graph:

Definition 3. *Let C be a biconnected component of a bipartite graph $G = (V_0, V_1; E)$. The* extension of C *with respect to vertices v_1 and v_2 in C and V_0, is a graph obtained from C by attaching leaves and pendant 2-paths to certain vertices in C. More specifically, if a vertex of C is adjacent to a leaf in G, then, in the extension of C, this vertex is adjacent to a leaf. If a vertex v in C has $d \geq 1$ non-leaf neighbors in G that do not belong to C, then:*

1. *If $v \neq v_1, v_2$ or $v \notin V_0$, then we attach d pendant 2-paths to v in the extension of C.*
2. *If $v = v_1 = v_2 \in V_0$ and $d \geq 2$, then we attach 2 pendant 2-paths to v in the extension of C.*
3. *Otherwise, we attach exactly one pendant 2-path to v in the extension of C.*

This definition is illustrated in Figure 4.

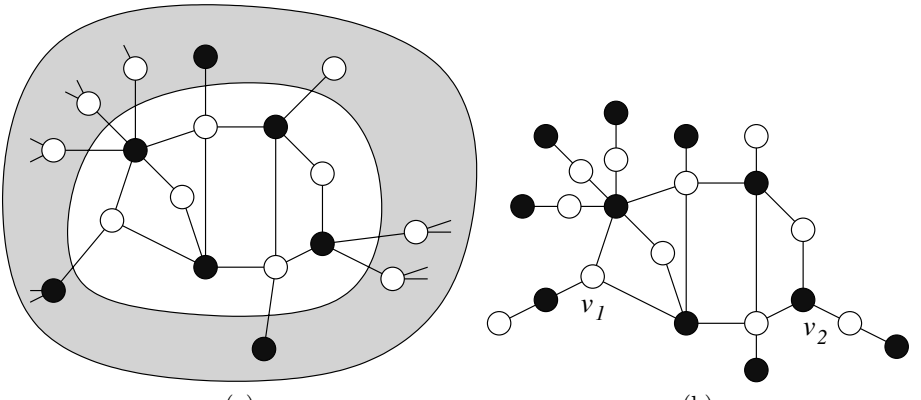


Fig. 4. (a) A biconnected component C in a bipartite graph $G = (V_0, V_1; E)$ where the darkened vertices belong to V_0, and (b) the extension of C with respect to v_1, v_2 and V_0

Definition 4. *Let C be a biconnected component containing at least three vertices in a bipartite graph $G = (V_0, V_1; E)$. Biconnected component C is* safe *with respect to V_0 if there exists a* safety certificate *for C with respect to V_0, namely at tuple $\langle v_1, v_2, P_1, P_2, \Psi_C \rangle$ consisting of two vertices v_1 and v_2 in C, two simple paths P_1 and P_2 in C_e, the extension of C with respect to v_1, v_2 and V_0, and a planar embedding Ψ_C of C, such that:*

1. *The vertices of P_1 and P_2 in C lie on the external face of Ψ_C;*
2. *P_1 and P_2 each contain a vertex of V_0 on each face of Ψ_C;*
3. *If a vertex of C belongs to V_0, then it belongs to P_1 or P_2 but not both;*
4. *If a vertex v in C has a neighbor outside C, then v belongs to P_1 or P_2;*
5. *If a vertex in C is a connecting vertex, then it is equal to v_1 or v_2;*
6. *Both v_1 and v_2 are end-vertices of the subpaths of P_1 or P_2 in C such that each path from v_1 to v_2 on the external face cycle of C in Ψ_C contains the subpath of P_1 in C or the subpath of P_2 in C;*
7. *If, for some vertex v in C, $G - v$ contains two components H_1 and H_2 vertex-disjoint with C that are not caterpillars, then $v = v_1 = v_2$; and*
8. *Each pendant 2-path in C_e belongs to P_1 or P_2.*

A safety certificate $\langle v_1, v_2, P_1, P_2, \Psi_C \rangle$ is said to be *tied* if $v_1 = v_2$. We note that the extension of Figure 4(b) does not have a safety certificate because one of the vertices has three pendant 2-paths; as a result, there are no paths P_1 and P_2 that contain all pendant 2-paths and each contain a vertex of V_0 on each face. Figure 5, however, shows that the biconnected component of Figure 4(a) is safe. The safety certificate showing this consists of the vertices labelled v_1 and v_2, the two highlighted paths P_1 and P_2, and the embedding of the component shown in the drawing.

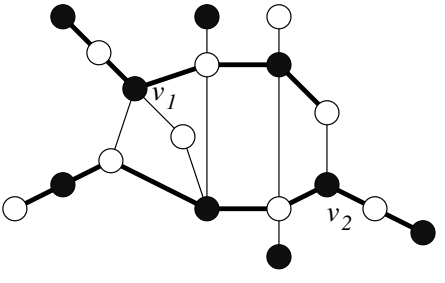


Fig. 5. The biconnected component of Figure 4(a) has a safety certificate $\langle v_1, v_2, P_1, P_2, \Psi \rangle$ where P_1 and P_2 are the highlighted paths

Based on these definitions, we state our characterization theorem:

Theorem 1. *A graph G is proper 3-layer planar if and only if G is bipartite and each vertex and each biconnected component of G is safe with respect to some bipartition class of G.*

We prove this theorem in two parts: we first prove the necessity of the conditions, and then we prove their sufficiency. The full proof is several pages long so we include only a sketch here.

To prove the necessity of the conditions, we consider a proper 3-layer planar drawing of a graph G. From the drawing, it is easy to see that G is bipartite. It is also easy though tedious to show that each vertex and each biconnected component is safe with respect to the bipartition class corresponding to the vertices drawn on the top and bottom layers. For example, to prove that a biconnected component C is safe, we must obtain the necessary safety certificate $\langle v_1, v_2, P_1, P_2, \Psi_C \rangle$. This is done first by selecting any simple path P from a leftmost to a rightmost vertex in the drawing. Letting the embedding Ψ_C of C be the embedding of C in the given drawing, we let v_1 be the first vertex of P in C, and v_2 be the last vertex of P in C. We obtain P_1 and P_2 by starting with two subpaths of the external face cycle of Ψ_C between v_1 and v_2, and then extending them as necessary to contain any pendant 2-paths in the extension of C. It is then straightforward though tedious to prove that $\langle v_1, v_2, P_1, P_2, \Psi_C \rangle$ is a safety certificate for C.

To prove the sufficiency of the conditions, we consider a graph that is bipartite, and for some bipartition class, each vertex is safe and each biconnected component has a safety certificate. Using this information, we show how to construct a proper 3-layer planar drawing of the graph. We first construct a special path P in the graph that contains each connecting vertex and, for each biconnected component C_i with safety certificate $\langle v_1^i, v_2^i, P_1^i, P_2^i, \Psi_{C_i} \rangle$, P also contains v_1^i and v_2^i. We then obtain the drawing as follows:

(a) We draw each biconnected component C_i according to the embedding Ψ_{C_i} in its safety certificate and insert each of these drawings into the main drawing in the order that they appear in P.
(b) We draw the remaining subpaths of P that connect consecutive biconnected components.
(c) Finally, we insert drawings of the pendant trees that are attached to vertices already drawn.

This completes our sketch of the proof of Theorem 1.

Testing whether or not a given graph is proper 3-layer planar is simply an application of the characterization given in Theorem 1. Many of the definitions of safety, both for vertices and biconnected components, depend on knowing whether or not various subgraphs of the input graph G are caterpillars. This can be computed in linear time by first computing a block-cut tree of G [14] and then applying a dynamic programming algorithm to the block-cut tree. A *block-cut tree* of a graph is a tree whose vertices correspond to cut-vertices and biconnected components in the graph. Edges in the block-cut tree connect biconnected components with the cut-vertices that they contain in the graph. We call the resulting algorithm IsProper3LayerPlanar:

Theorem 2. *Algorithm* IsProper3LayerPlanar *determines whether or not a graph is proper* 3*-layer planar in linear time.*

We can transform this recognition algorithm into a drawing algorithm by returning a proper 3-layer planar drawing anytime the recognition algorithm returns TRUE. The drawing is constructed as described in the sufficiency proof of Theorem 1. To do this, we require the set of connecting vertices and a safety certificate for each biconnected component. Fortunately, as described above, the recognition algorithm already computes these things.

5 Conclusions

We have shown how to determine whether or not a graph is proper 3-layer planar in linear time, and, if it is, we show how to obtain a proper 3-layer planar drawing in the same asymptotic running time. It is not too difficult to see that our basic approach is identical to that of Fößmeier and Kaufmann [6]. For example, we observe that the path mentioned in our algorithm actually contains all vertices that trigger recursive calls in their algorithm. Unfortunately, as can be seen from the length of our characterization statement, the effort required to obtain correct algorithms for three layers is much greater than for two layers. Therefore, we believe that a new approach will be required to obtain efficient algorithms for four or more layers.

One possible approach involves applying graph operations that reduce the graph to an empty graph if and only if the graph has a planar drawing on a given number of layers. Arnborg and Proskurowski [1] use this type of approach to recognize graphs of treewidth three, and Matousek and Thomas [11] modify their set of reductions to obtain an efficient quadratic-time recognition algorithm. Generalizing these results, Sanders [12] shows that this approach can be used to efficiently recognize graphs with treewidth four. Our hope is to similarly find a set reductions for proper 3-layer planar graphs and likewise use them to obtain reductions for proper 4-layer planar graphs. We believe that this approach might be successful because Dujmović *et al.* [4] use pathwidth, a restricted version of treewidth, to obtain algorithms for recognizing k-layer planar graphs (inefficient algorithms though they may be). In addition to this, we show in [13] how to use pathwidth to obtain planar drawings of trees on an minimum number of layers. These results show that pathwidth and hence treewidth are closely related to the number of layers in planar layered graph drawings.

Acknowledgements

I thank my supervisor Sue Whitesides for introducing me to this problem at one of her workshops, for double-checking the correctness of my results, and for many helpful suggestions for improving the presentation of this paper. I also thank Vida Dujmović for productive discussions about the algorithm of Fößmeier and Kaufmann.

References

1. Arnborg, S., Proskurowski, A.: Characterization and recognition of partial 3-trees. SIAM Journal of Algebraic and Discrete Methods **7** (1986) 305–314
2. Battista, G.D., Eades, P., Tamassia, R., Tollis, I.G.: Graph Drawing: Algorithms for the Visualization of Graphs. Prentice-Hall (1999)
3. Cornelsen, S., Schank, T., Wagner, D.: Drawing graphs on two and three lines. In Goodrich, M., Kobourov, S., eds.: Graph Drawing, 10th International Symposium (GD 2002). Volume 2528 of Lecture Notes in Computer Science., Springer-Verlag (2002) 31–41
4. Dujmović, V., Fellows, M.R., Hallett, M.T., Kitching, M., Liotta, G., McCartin, C., Nishimura, N., Ragde, P., Rosamond, F.A., Suderman, M., Whitesides, S., Wood, D.R.: On the parameterized complexity of layered graph drawing. In auf der Heide, F.M., ed.: Algorithms, 9th European Symposium (ESA 2001). Volume 2161 of Lecture Notes in Computer Science., Springer-Verlag (2001) 488–499
5. Eades, P., McKay, B., Wormald, N.: On an edge crossing problem. In: Proceedings of the 9th Australian Computer Science Conference, Australian National University (1986) 327–334
6. Fößmeier, U., Kaufmann, M.: Nice drawings for planar bipartite graphs. In Bongiovanni, G.C., Bovet, D.P., Battista, G.D., eds.: Proceedings of the 3rd Italian Conference on Algorithms and Complexity (CIAC 1997). Volume 1203 of Lecture Notes in Computer Science., Springer-Verlag (1997) 122–134
7. Harary, F., Schwenk, A.: A new crossing number for bipartite graphs. Utilitas Mathematica **1** (1972) 203–209
8. Heath, L.S., Rosenberg, A.L.: Laying out graphs using queues. SIAM Journal on Computing **21** (1992) 927–958
9. Kaufmann, M., Wagner, D.: Drawing Graphs: Methods and Models. Volume 2025 of Lecture Notes in Computer Science. Springer-Verlag (2001)
10. Lengauer, T.: Combinatorial Algorithms for Integrated Circuit Layout. Wiley (1990)
11. Matousek, J., Thomas, R.: Algorithms finding tree-decompositions of graphs. Journal of Algorithms **12** (1991) 1–22
12. Sanders, D.P.: On linear recognition of tree-width at most four. SIAM Journal on Discrete Mathematics **9** (1995) 101–117
13. Suderman, M.: Pathwidth and layered drawings of trees. International Journal of Computational Geometry & Applications **14** (2004) 203–225
14. Tarjan, R.: Depth-first search and linear graph algorithms. SIAM Journal on Computing **1** (1972) 146–160
15. Tomii, N., Kambayashi, Y., Yajima, S.: On planarization algorithms of 2-level graphs. Technical Report EC77-38, Institute of Electronic and Communication Engineers of Japan (IECEJ) (1977)
16. Warfield, J.N.: Crossing theory and hierarchy mapping. IEEE Transactions on Systems, Man, and Cybernetics **7** (1977) 502–523
17. Waterman, M.S., Griggs, J.R.: Interval graphs and maps of DNA. Bulletin of Mathematical Biology **48** (1986) 189–195