On Rectangle Visibility Graphs

Prosenjit Bose[1], Alice Dean[2], Joan Hutchinson[3], and Thomas Shermer[4] *

[1] Université du Québec à Trois-Rivières
[2] Skidmore College
[3] Macalester College
[4] Simon Fraser University

Abstract. We study the problem of drawing a graph in the plane so that the vertices of the graph are rectangles that are aligned with the axes, and the edges of the graph are horizontal or vertical lines-of-sight. Such a drawing is useful, for example, when the vertices of the graph contain information that we wish displayed on the drawing; it is natural to write this information inside the rectangle corresponding to the vertex. We call a graph that can be drawn in this fashion a *rectangle-visibility graph*, or *RVG*. Our goal is to find classes of graphs that are RVGs.
We obtain several results:

1. For $1 \leq k \leq 4$, k-trees are RVGs.
2. Any graph that can be decomposed into two caterpillar forests is an RVG.
3. Any graph whose vertices of degree four or more form a distance-two independent set is an RVG.
4. Any graph with maximum degree four is an RVG.

Our proofs are constructive and yield linear-time layout algorithms.

1 Introduction

In this paper we consider the problem of drawing a graph in the plane so that the vertices of the graph are drawn as rectangles and the edges are horizontal or vertical line segments. We are in particular interested in drawings where each of the line segments can be thickened to have positive width, and none of these thickened segments (which we call *bands of visibility*) intersects the interior of any of the rectangles, although the bands may themselves cross or intersect. We call a graph a *rectangle-visibility graph* (or *RVG* for short) if it has such a drawing; we call the drawing itself the *layout* of the graph.

We study three variations on this idea. In the first, we require that the graph be drawn so that every pair of rectangles with a possible band of visibility between them represents a pair of vertices joined by an edge in the graph, and furthermore that no two rectangles have sides contained in the same (horizontal or vertical) line; we call such graphs *noncollinear RVGs*. In the second, we still

* Supported by NSERC Canada Reseach Grant OGP0046218, and DIMACS, Rutgers University, funded by NSF under contract STC–91–19999 and the New Jersey Commission on Science and Technology.

require that every possible visibility band represents an edge, but we allow two rectangles to have collinear edges; we call such graphs *collinear RVGs*. In the third, we do not require that every possible visibility band represents an edge; we call such graphs *weak RVGs*. Collinearities in drawings of weak RVGs can be eliminated by perturbation.

The results that we establish in this paper are that certain classes of graphs are RVGs (of the different types described above). We will establish that:

1. For $k = 1$ or 2, partial k-trees are noncollinear RVGs.
2. For $1 \leq k \leq 4$, k-trees are noncollinear RVGs (and thus partial 3-trees and partial 4-trees are weak RVGs).
3. All graphs that can be decomposed into two forests of caterpillars are noncollinear RVGs.
4. All graphs whose high-degree vertices form a distance-two independent set are weak RVGs.
5. All graphs whose high-degree vertices form a distance-three independent set are collinear RVGs.
6. All graphs whose high-degree vertices form a distance-four independent set (with a slight extra condition) are noncollinear RVGs.
7. All graphs whose maximum vertex degree is three are noncollinear RVGs.
8. All graphs whose maximum vertex degree is four are weak RVGs.

A caterpillar is a tree containing a path with the property that every vertex is at distance at most one from the path. A high-degree vertex is a vertex of degree four or more.

The problem that we are studying has application to a type of VLSI design known as *two-layer routing*. In two-layer routing, one embeds processing components and their connections (sometimes called *wires*) in two layers of silicon (or other VLSI material). The components are embedded in both layers. The wires are also embedded in both layers, but one layer holds only horizontal connections, and the other holds only vertical ones. If a connection must be made between two components that are not cohorizontal or covertical, then new components (called *vias*) are added to connect horizontal and vertical wires together, resulting in bent wires that alternate between the layers. However, vias are large compared to wires and their use should be minimized. In this setting, asking if a graph is a rectangle-visibility graph is the same as asking if a set of components can be embedded so that there is a two-layer routing of their connections that uses no vias. Our requirement that visibility bands have positive width is motivated by the physical constraint that wires must have some minimum width. A similar problem arises in printed-circuit board design, as printed-circuit boards naturally have two sides, and connecting wires from one side to the other (the equivalent of making vias) is relatively expensive [6].

The motivational two-layer problem discussed above abstracts away one feature of most two-layer routing problems: the processing components are often of a specific size. In other words, not only is a graph representing the components and their connections given, but each vertex of the graph also has a specified width and height. We turn our attention to this consideration only briefly in this paper; we show that any graph with linear arboricity two (which includes graphs with maximum degree three), no matter what widths and heights are specified for the vertices, can be laid out as a rectangle-visibility graph where each rectangle is of the specified size.

Rectangle-visibility layouts of graphs are also of use in other areas of graph drawing. One common graph drawing problem is *labelling*: writing information about the vertices and edges of a graph on the drawing, without having labels intersect or overwrite either other labels or graph components. A common solution to the vertex-labelling problem is to draw each vertex as a region (such as a relatively large circle) inside of which the information is written. Rectangle-visibility layouts are particularly suited to this approach. Furthermore, if we have a graph for which we can construct a layout with specified rectangle size, as discussed above, we can dimension the rectangle to fit exactly around the information that we wish to write inside it. Rectangle-visibility layouts are also advantageous when we wish to edge-label a graph; all edges and vertex sides in such a layout are horizontal or vertical (as opposed to layouts where edges can have any slope or curve), so there is no complicated geometry involved in placing an edge label.

One final advantage of rectangle-visibility layouts is in the nature of its edge crossings. Any edge crossing is between a horizontal edge and a vertical edge, and therefore perpendicular; such crossings make it easier for the eye to follow an edge without being distracted by crossing edges. Furthermore, if we have a collinear or noncollinear RVG, then we can eliminate crossings altogether, by *not* drawing the edges. For drawings intended to be read by people, this approach is only feasible for small graphs, or graphs with relatively short edges.

In the remainder of this paper, we establish our results described above. In Section 2, we review what is known about RVGs and related graphs, and give formal definitions necessary for the problems we consider and the techniques that we use. In Section 3, we establish results about which k-trees and partial k-trees are RVGs. In Section 4, we show that graphs that can be decomposed into two caterpillar forests are noncollinear RVGs; this class includes graphs with linear arboricity two, or with maximum degree three. In that section, we also discuss a few extensions of the caterpillar forest result. In Section 5, we present two geometric lemmas that state that graphs that have been decomposed in a certain way are rectangle-visibility graphs. In Section 6, we use the lemmas of Section 5 to show that graphs whose high-degree vertices are "far enough apart" are RVGs. In Section 7, we show that maximum-degree four graphs are weak RVGs; the main argument is an extension of one of the lemmas of Section 5. Finally, in Section 8, we draw our conclusion and discuss future directions for this work.

In this extended abstract, we omit the proofs of several of our results without notice; these proofs appear in the full version of our work [2, 10].

2 Background and Definitions

2.1 Bar-visibility graphs

The study we pursue on rectilinear drawings of graphs in the plane began with Luccio, Mazzone, and Wong [9] and Duchet *et al.* [5] in their studies of horizontal lines in the plane and vertical visibilities between them. Following the terminology of Tamassia and Tollis [12], we call a graph G a *bar-visibility graph* or *BVG* if its vertices can be represented by closed horizontal line segments in the plane, pairwise disjoint except possibly for overlapping endpoints, in such a way that two vertices u and w are adjacent if and only if each of the corresponding segments is *vertically visible* from the other. The vertices u and w are called vertically visible if there is a non-degenerate rectangular region $B_{u,w}$ (the *band of visibility for u and w*) with two opposite sides that are subsets of each of these segments, and $B_{u,w}$ intersects no other segment. A set of segments realizing a BVG is called a *layout* of the BVG.

Note that this definition *does* require non-degenerate visibility bands, and it does *not* require noncollinearity of segments. Bar-visibility graphs are easily seen to be planar; in addition they have been characterized by Wismath [13], and independently by Tamassia and Tollis [12], as those planar graphs that can be drawn in the plane with all cut-vertices on a single face. The question of whether a graph has a bar-visibility layout can be decided in linear time [12]. Figure 1 shows a bar-visibility layout of the 4-cycle and a bar-visibility layout of a forest containing two trees. If each bar in a bar-visibility layout is widened vertically to become a rectangle, we obtain a rectangle-visibility layout, but more can be achieved if both vertical and horizontal visibility is employed, as explained below.

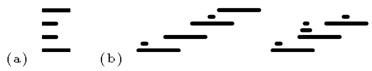

Fig. 1. A collinear bar-visibility layout of C_4, and a bar-visibility layout of a forest.

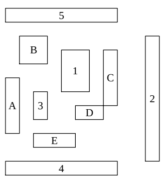

Fig. 2. A rectangle-visibility layout of $K_{5,5} + e$.

2.2 Rectangle-visibility graphs

Consider a collection $\mathcal{R}$ of rectangles in the plane, where each rectangle has its sides parallel to the axes, and rectangles may share boundary points but not interior points. In this situation, we will call two rectangles u and v *visible* if there is a *band of visibility* $B_{u,v}$ between them: $B_{u,v}$ is a rectangular region with two opposite sides that are subsets of u and v, and such that $B_{u,v}$ intersects no other rectangle of $\mathcal{R}$. The *visibility graph* of $\mathcal{R}$ is the graph of the visibility relation on the elements of $\mathcal{R}$. We call a graph G a *rectangle-visibility graph* or RVG if it is the visibility graph of some collection $\mathcal{R}$ of rectangles; in this situation, $\mathcal{R}$ is called a *layout* of G. The edges of a rectangle-visibility graph G can be partitioned into the two sets representing horizontal and vertical visibility; each of these two edge sets forms a BVG. Thus G, as a union of two planar graphs, is said to have *thickness-two*. Much less is known about thickness-two graphs than about planar ones, although their recognition is known to be NP-complete.

Wismath [14] has shown that every planar graph has a rectangle-visibility layout. Hutchinson, Shermer, and Vince [8] show that a rectangle-visibility graph with n vertices has at most $6n-20$ edges, in contrast with thickness-two graphs, which may have at most $6n-12$ edges; in both cases these bounds are attainable. Dean and Hutchinson [4] show that $K_{5,5}$ is *not* a rectangle-visibility graph though $K_{5,5}$ plus any edge is; see Figure 2. Thus rectangle-visibility graphs are not closed under the formation of subgraphs.

Each of the classes of BVGs and RVGs has two important subclasses: graphs with *noncollinear* layouts and those with *strong* layouts, as defined below.

2.3 Collinear and noncollinear layouts

A bar-visibility layout is called *noncollinear* if no two line segments have collinear endpoints; a rectangle-visibility layout is noncollinear if no two rectangles have collinear sides. The 4-cycle (see Figure 1a) does not have a noncollinear bar-visibility layout; Figure 3 shows a (collinear) rectangle-visbility layout of $K_{4,4}$ minus an edge; by a result in [3] it has no noncollinear layout, but a noncollinear layout of $K_{4,4}$ minus two edges is shown in Figure 10.

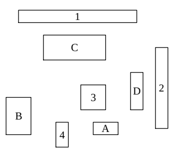

Fig. 3. A collinear (and strong) layout of $K_{4,4}-e$.

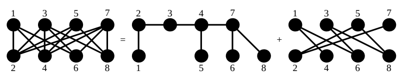

Fig. 4. $K_{4,4}$ minus two edges decomposed into two caterpillars.

2.4 Strong and weak layouts

G is a *strong bar-visibility graph* if its vertices can be represented by *disjoint* closed horizontal line segments in the plane in such a way that two vertices u and w are adjacent if and only if each of the corresponding segments is vertically visible from the other, with visibility permitted along *degenerate* rectangles, i.e., along lines. Similarly, G is a *strong rectangle-visibility graph* if its vertices can be represented by disjoint closed rectangles in the plane, with sides parallel to the axes, in such a way that two vertices u and w are adjacent if and only if each of the corresponding rectangles is vertically or horizontally visible from the other, again with visibility permitted along lines.

It is easy to see that noncollinear bar-visibility graphs are a subclass of strong bar-visibility graphs, which are in turn a subclass of bar-visibility graphs; analogous inclusions hold in the case of rectangle-visibility graphs. Tamassia and Tollis [12] show that this subclass ordering for bar-visibility graphs is strict. In [3] Dean and Hutchinson conjecture that the analogous subclass ordering for rectangle-visibility graphs is also strict, and they show that noncollinear rectangle-visibility graphs form a strict subclass of strong rectangle-visibility graphs. Figure 3 shows a strong layout of a graph that has no noncollinear layout.

Another class of BVGs and RVGs that can be considered is *weak* BVGs and RVGs. A graph is a weak BVG (respectively, weak RVG) if it is a subgraph of some BVG (respectively, RVG). This usage of *weak* and *strong*, which is not antonymous, was introduced by Tamassia and Tollis [12]. In [5] it is shown that every planar graph is a weak BVG, hence the containment from BVG to weak BVG is strict. This containment is also strict for RVGs, for example, from the fact that $K_{5,5}$ is not an RVG but the addition of an edge creates an RVG (see Figure 2).

2.5 Decomposition into trees, caterpillars, and paths

Earlier we noted that the edges of an RVG can be partitioned to form two BVGs. More generally, we will say that graph G can be *decomposed* into graphs $G_1, G_2, \ldots G_k$ if all of $G, G_1, G_2, \ldots G_k$ have the same vertex set and the edge set of G is exactly the union of the edge sets of $G_1, G_2, \ldots G_k$ (which must be disjoint). This can be viewed as coloring the edges of G with k different colors so that one color class forms G_1, one forms G_2, etc. We will use the concepts of decomposition and edge-coloring interchangably; we may, for instance, once we have completed an edge-two-coloring of a graph, say that we have decomposed that graph into two subgraphs. Figure 4 shows a decomposition of $K_{4,4}$ minus two edges into two caterpillars.

When we two-color, we will call the color classes red and blue (and say that red and blue are opposite colors). We will call a cycle monochromatic if all edges of the cycle are the same color. We will call a vertex monochromatic if all incident edges are the same color; vertices of degree zero are considered trivially monochromatic.

Recall that a *caterpillar* is defined as a tree containing a simple path $P(a,b)$ such that every vertex not on $P(a,b)$ is distance one from $P(a,b)$. A vertex on $P(a,b)$ will be called a *path vertex*, and one not on $P(a,b)$ will be called a *foot vertex*. Similarly, we will call an edge that connects a path vertex to a foot vertex a *leg*, and one connecting two path vertices a *path edge*. Figures 5a and 5b show trees that, respectively, are and are not caterpillars. A *linear forest* (respectively, a *caterpillar forest*) is a forest, each of whose components is a simple path (respectively, a caterpillar). The *arboricity* (respectively, *linear arboricity, caterpillar arboricity*) of a graph G is the minimum k such that G can be decomposed into $G_1, G_2, \ldots, G_k$, where each G_i is a forest (respectively, a linear forest, a caterpillar forest). Thus Figure 4 shows that $K_{4,4}$ minus two edges has caterpillar arboricity two. Clearly a linear forest is a caterpillar forest, and we show that a graph with caterpillar arboricity at most two is an RVG. See Figure 10 for a layout of the graph of Figure 4.

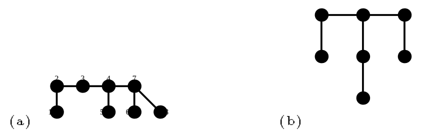

Fig. 5. (a) A graph that is a caterpillar, an interval graph, a 1-tree, and a noncollinear BVG. (b) A graph that is neither a caterpillar nor an interval graph, but is a 1-tree and a noncollinear BVG.

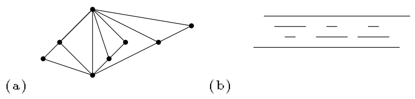

Fig. 6. A 2-tree and its bar-visibility layout.

We use $G \setminus E(H)$ to denote the graph G with the edges of subgraph H removed, and $G \setminus v$ to mean the graph G with the vertex v removed.

It is known [7] that graphs with maximum degree three have linear arboricity two. We present a new proof of this result that leads to a simple, easily-implementable algorithm for finding this decomposition. Then we find a rectangle-visibility layout for these graphs with a property of particular interest to practitioners of graph drawing: a rectangle layout can be constructed where the dimensions of the rectangle corresponding to each vertex is prespecified. For example, this allows each rectangle to be sized to fit exactly around text or information to be inscribed.

3 k-Trees and Partial k-Trees

A *k-tree* is either a k-vertex complete graph, or a graph formed from another k-tree T by finding a k-vertex clique K of T and adding a new vertex that is adjacent to every vertex in K (and no others). Thus, trees are the same as 1-trees, and every maximal outerplanar graph is a 2-tree; a 2-tree is shown in Figure 6. A *partial k-tree* is a subgraph of a k-tree. For example, partial 1-trees are the same as forests, and every series-parallel graph is a partial 2-tree. (There are at least two nonequivalent definitions of series-parallel graphs in the literature; one of these is equivalent to partial 2-trees and the other is a subclass of partial 2-trees.) A partial 2-tree is shown in Figure 7a; this graph is not a collinear BVG but it is a noncollinear RVG.

The following two theorems can be easily derived from the result of [9].

Theorem 1. *Every 1-tree and 2-tree is a noncollinear bar-visibility graph.*

Theorem 2. *Every partial 1-tree (forest) is a noncollinear bar-visibility graph.*

As every BVG is an RVG, the two previous theorems hold for RVGs as well. In fact, we can extend both of them for RVGs, showing that every k-tree ($1 \leq k \leq 4$) is a noncollinear RVG, as is every partial k-tree ($k = 1$ or 2).

Theorem 3. *For $1 \leq k \leq 4$, every k-tree is a noncollinear rectangle-visibility graph.*

The range of k in this theorem cannot be increased, as there are 5-trees containing $K_{5,13}$, which has thickness three.

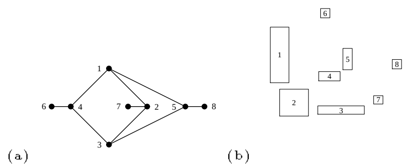

Fig. 7. A partial 2-tree and its rectangle-visibility layout.

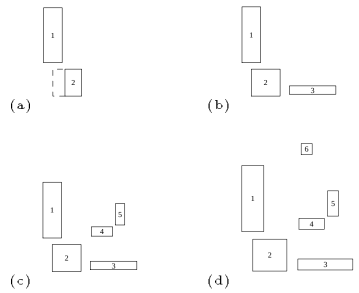

Fig. 8. The construction of the layout of Figure 7.

Theorem 4. *Every partial 2-tree is a noncollinear rectangle-visibility graph.*

Proof. Let G be a partial 2-tree, and H be the 2-tree that it is a subgraph of. We proceed by induction on the number of vertices of H.

Let a rectangle R in a layout be called *N-visible* if it can see "to infinity" in a northward direction. We call the unbounded band of visibility from R northward a *band of N-visibility*, and use *E-visible* and *band of E-visibility* for the eastward direction.

In laying out G, we maintain the invariant that there are no collinearities, each rectangle is both N-visible and E-visible, and if two vertices form a clique of the underlying 2-tree, then the band of N-visibility for one intersects the band of E-visibility for the other. The basis, when H has two vertices, is easily handled.

In the general case, let v be a vertex that we can remove from H to get another 2-tree. The graph G' that is formed by removing v from G is a partial 2-tree, and we inductively construct a layout of G' that satisfies the invariant.

Consider reintroducing v to re-form G. If v is not a vertex of G, then we do nothing. Otherwise, the vertex v is connected to two vertices w and x in H. By the invariant, one of the rectangles corresponding to these vertices, wlog $R(x)$, has a band of N-visibility that intersects the band of E-visibility of the other ($R(w)$). We let the intersections of these bands be called WX.

Each of the edges vw and vx may be present or absent in G; we need to show that we can place $R(v)$ so that the correct ones of these edges are present and our invariant is maintained. We examine four cases, omitting the analysis.

Case 1: The edges vw and vx are both present in G. Let $R(v)$ be placed inside the bottom-left quarter of WX so as not to introduce any collinearities; see the placement of rectangle 4 (or rectangle 5) in Figure 8c for an example.

Case 2: The edge vw is present in G, but vx is not. Let T be the line one unit above the top of the rectangle with the highest top in the layout. We place $R(v)$ so that its bottom is contained in T, it is one unit high, and it is contained in the left half of the band of N-visibility from $R(w)$; see the placement of rectangle 6 in Figure 8d for an example.

Case 3: The edge vx is present in G, but vw is not. This case is symmetric to Case 2.

Case 4: Neither vw nor vx is present in G. Let T be a line above the top rectangle as in Case 2, and L be a similar line one unit to the left of the leftmost rectangle. We let $R(v)$ be a unit square whose bottom is contained in T and whose right side is contained in L.

In all cases, we have shown how to embed $R(v)$, and the theorem follows by induction. $\square$

We note here that Biedl [1] has concurrently and independently proven that all series-parallel graphs are RVGs; depending on her definition of series-parallel, her result is either subsumed by or equivalent to our theorem.

Figure 7b shows the construction of the last proof applied to the graph of Figure 7a. This result cannot be extended to partial 3-trees, as $K_{3,5}$ is a partial 3-tree but is not a noncollinear RVG [4]. Theorem 3 implies that partial 3-trees are weak RVGs; the question of whether or not they are collinear RVGs is still open.

4 Caterpillar Forests

Recall that a graph G is called an *interval graph* if each vertex can be represented by an interval on the real line so that two vertices of G are adjacent if and only if the corresponding intervals have nonempty intersection; we call such a representation an *interval layout* of G. For any interval graph, the intervals in the layout can be chosen so that no two share an endpoint. Figure 9 shows an interval layout of the caterpillar shown in Figure 5a.

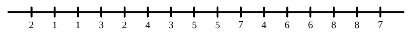

Fig. 9. An interval layout of the graph of Figure 5a.

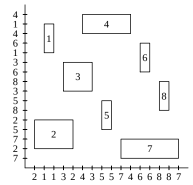

Fig. 10. A rectangle-visibility layout of the graph of Figure 4.

If each horizontal segment (bar) of a bar-visibility layout of a graph G is vertically projected to an interval on the x-axis, the resulting interval graph is a supergraph of G. Similarly, a rectangle-visibility layout can be projected both vertically and horizontally to obtain two interval graphs, supergraphs of the vertical and horizontal BVGs. The idea of the rectangle-visibility layout constructions in this section is to reverse this process, starting with two interval graphs that are *exactly* the horizontal and vertical BVGs, and combining their interval layouts to obtain a rectangle-visibility layout.

Theorem 5 (The Caterpillar Theorem). *Every graph with caterpillar arboricity two is a noncollinear RVG.*

Proof. We give only the construction.

An arbitrary caterpillar C contains a path $P(x_0, x_k) = x_0, x_1, \ldots, x_k$ of path vertices; to lay out C, we lay $P(x_0, x_k)$ out on an axis with x_i represented by the interval $(2i, 2i+3)$, $i = 0, 1, \ldots, k$. We then lay out the set of foot vertices adjacent to each path vertex x_i as consecutive disjoint subintervals of $(2i+1, 2i+2)$; see Figure 9. A caterpillar forest can then be laid out by putting the interval layouts for the caterpillar forest's components one after another.

Suppose that G can be decomposed into caterpillar forests F_1 and F_2. Lay out F_1 and F_2 as interval graphs on the x-axis and y-axis, respectively, as described above. For each vertex v of G, let $R(v)$ be the rectangle that is the cartesian product of the two intervals (one on the x-axis and one on the y-axis) that correspond to v in the layouts of F_1 and F_2. This gives a layout of G; see Figure 10 for an example. □

In subsequent work, Shermer [11] has shown that determining if a graph has caterpillar arboricity two is NP-complete, and we suspect that the other decomposition problems are also NP-complete. However, it is still possible that some natural, easily-recognizable graph classes are subclasses of caterpillar arboricity two. The graphs with maximum degree three form one such class; any such graph has linear arboricity two [7]. In the full paper, we present a new, algorithmically-useful proof of this fact. This establishes the following theorem:

Theorem 6. *Every maximum-degree 3 graph is a noncollinear RVG.*

Consider the interval layout that we have specified for a path. By making the overlap of consecutive intervals small enough, we can contract and expand the center portion of any interval (moving the rest of the layout) until that interval has a desired length. Applying this to all intervals, we see that we can lay out a path as an interval graph where each vertex of the path has the length of its interval prespecified.

We can obviously extend this idea to linear forests. This implies that we can also prespecify the interval sizes for the two linear forest layouts used in the layout construction algorithm (the proof of Theorem 5) for graphs with linear arboricity two. We can use this, for instance, to create a layout of all unit squares, or to size each rectangle to fit exactly around text or graphics that we wish to inscribe in it.

5 Technical Extensions of the Caterpillar Theorem

In this section, we present two technical lemmas that are extensions of the Caterpillar Theorem (Theorem 5). These lemmas will be used in the proof of the results in the next section, and contain the essential geometry for those results; the proofs in the next section are mainly graph-theoretic.

Lemma 7. *Let G be a graph such that it can be decomposed (edge-colored) into two graphs whose connected components are either caterpillars or cycles, and furthermore, no two cycles of different colors have more than one vertex in common. The graph G is a weak RVG, and a weak layout of G can be constructed in linear time.*

Proof. We proceed somewhat as in the proof of the Caterpillar Theorem, laying out each colored subgraph as intervals on a line, and taking the cartesian product for each vertex. Caterpillars are laid out on the line as before, and cycles are first laid out so that all vertices of the cycle have exactly the same interval (and this interval overlaps no other intervals). Figure 11a shows a five-cycle and its interval layout.

Before actually taking the cartesian products to form rectangles, we adjust the intervals for the vertices of each cycle. Consider a cycle laid out as intervals on the horizontal line. If we were to take the cartesian products for the vertices in this cycle, there would be exactly one with the highest top edge. In Figure 11b, example cartesian products are shown; the vertex e has the highest top edge.

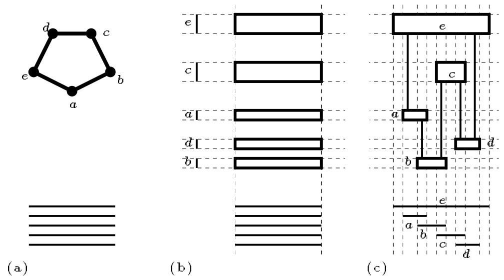

Fig. 11. Laying out a cycle.

We leave the interval for this top vertex t as it is, but lay out the rest of the cycle as a path (legless caterpillar) that starts after t does, and ends before t does; an example is shown on the bottom of Figure 11c. This effectively replaces the cycle in the horizontal layout by a cycle with all of the chords to one of its vertices. Unfortunately, we are not yet finished. These new chords may duplicate edges found in the vertical layout, so we will remove these edges from that layout. The details of this adjustment are omitted here.

Having applied this construction to each cycle, we can now take the cartesian products of the two intervals for each vertex. The proof that we now have a weak layout is straightforward and omitted. □

Let v be a path vertex on a caterpillar, and V' be the adjacent foot vertices. We can enlarge the caterpillar by adding to it a linear forest induced by the vertices of V', and repeating this for each path vertex v in the caterpillar. We call a graph formed in this manner a *caterpillar with footpaths.* A caterpillar with footpaths is shown in Figure 12. We allow the linear forests that we add to have no edges; i.e. a caterpillar is considered to be a caterpillar with footpaths. Each path induced by the vertices adjacent to a path vertex v is called a *footpath*, and is said to have *apex* v. In Figure 12, $bcde$ is a footpath with apex a.

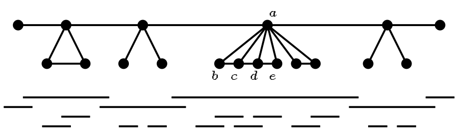

Fig. 12. A caterpillar with footpaths and its interval layout

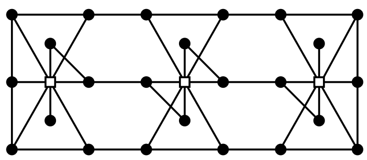

Fig. 13. A 2-hilly graph.

Suppose that we have a graph G whose edges have been colored so that each color class forms connected components that are caterpillars with footpaths. In this situation, we will call a footpath of one color *safe* if all of its vertices lie in one connected component C of the other color, and furthermore the apex of the footpath is not in C. This condition ensures that the cartesian product construction keeps the rectangles corresponding to the vertices of the footpath together, all on one side of the rectangle corresponding to the apex. A *caterpillar with safe footpaths* is then a caterpillar with footpaths where each footpath is safe.

Lemma 8. *Let G be a graph that can be decomposed into two graphs whose connected components are either caterpillars with safe footpaths or triangles. The graph G is a noncollinear RVG, and a noncollinear layout of G can be constructed in linear time.*

6 k-Hilly graphs

Recall that we call a vertex with degree four or more a *high-degree* vertex; we correspondingly call a vertex with degree three or less a *low-degree* vertex. We call a graph *k-hilly* if the high-degree vertices form a distance-k independent set; i.e. there is no path of length k or less starting at a high-degree vertex and ending at another one. A 4-hilly graph is shown in Figure 13; the high-degree vertices are shown as white squares (a convention that we will follow throughout the section on k-hilly graphs). We will also use a slightly more restrictive version of k-hilly; we call a graph k-hilly* if it contains no path of length k or less that starts and ends at high-degree vertices. This differs from k-hilly in that it also forbids cycles of length k or less that contain a high-degree vertex.

Theorem 9. *Every 2-hilly graph G is a weak RVG, and a weak layout of G can be constructed in linear time.*

Proof. The following algorithm constructs a layout of G as a weak RVG. The bulk of the algorithm two-colors the edges of G so as to be able to apply Lemma 7. The main idea is in Steps 1, 7, 8, and 13: we remove the high-degree vertices, color the remaining graph so that each color class is a linear forest and each vertex that used to be adjacent to a high-degree vertex is monochromatic, and then reintroduce the high-degree vertices, coloring their edges opposite the color of their monochromatic low-degree end. If this scheme worked perfectly, each color class would then be a collection of paths (through low-degree vertices) and stars (one centered at each high-degree vertex). However, the coloring is not quite possible as stated–but it can be done if cycles are allowed in each color class. The remainder of the steps are for winnowing out and dealing with the cases where cycles are necessary. Some details are omitted in this extended abstract.

Step 1. Remove high-degree vertices. Construct a graph G' from G by removing all high-degree vertices and the edges incident on them. We will use the term *reduced vertex* to refer to a vertex that is present in both G and G' but has smaller degree in G' than it did in G. A reduced vertex has degree at most two in G'.

Step 2. Remove cycles of reduced vertices. Remove all cycles of reduced vertices from G'. Note that such a cycle necessarily forms a connected component of G'.

Step 3. Remove near-cycles of reduced vertices. If there are any cycles in G' that consist of exactly one non-reduced vertex a and several reduced vertices, remove all the elements of these cycles except the vertices a from G'.

Step 4. Contract paths of reduced vertices. Contract all of the edges of any path of reduced vertices in G'. The vertex that such a path contracts to will also be called a reduced vertex. This procedure does not create any doubled edges, as the situations leading to doubled edges were removed in Step 3.

At this point, there are no two adjacent reduced vertices in G'.

Step 5. Contract unbridged reduced vertices. Let v be a reduced vertex with two neighbors c and d. We will call v *bridged* if cd is an edge of G', and *unbridged* otherwise.

Repeat the following procedure until no unbridged reduced vertices remain in G': First, find an unbridged reduced vertex v with neighbors c and d. Then, contract the edge vc (i.e., remove vc, vd, and d, and replace with the edge cd).

Step 6. Set aside Θ-components. We will call a connected component of G' a *Θ-component* if it is a four-cycle $vcwd$ with chord cd and reduced vertices v and w. We let G_2 be the subgraph of G' that consists of all of the Θ-components, and G_1 be the remaining components of G'.

Now, each pair of degree-2 reduced vertices in G_1 has distance at least three.

Step 7. Color G_1 as two linear forests.

Step 8. Adjust coloring to make reduced vertices monochromatic. We want a coloring of G_1 where each reduced vertex is incident on edges of only one color. If a reduced vertex has degree zero or one, then it is monochromatic in any coloring of the edges; thus we are concerned only with degree-two reduced vertices.

Find a bichromatic reduced vertex v in G'. By Step 5, v, along with its two adjacent vertices c and d, forms a triangle. If c has degree three, then let b be its third neighbor; similarly, if d has degree three, let e be d's third neighbor. The situation is shown in Figure 14a. Neither b nor e can be a degree-2 reduced vertex, as noted in Step 6.

In the coloring of G_1, the triangle cdv has two edges of one color, and one of the other color. Wlog we may assume that the edges cv and cd have the same color, and that color is red, and that dv is blue. If bc is present, it must

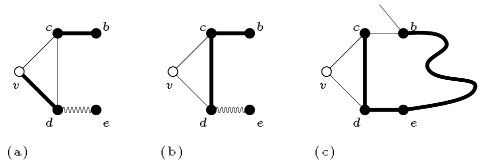

Fig. 14. Making a reduced vertex monochromatic.

be blue, and de, if present, may be either color. This coloring is also shown in Figure 14a. It now must be the case that either the recoloring of Figure 14b or Figure 14c leaves the coloring as a decomposition into two linear forests, with v monochromatic. Note that the only edges not incident on v that change color are cd and possibly cb. Neither of these edges is incident on a degree-2 reduced vertex, and thus we have not changed any reduced vertex from monochromatic to bichromatic.

We have reduced the number of bichromatic reduced vertices in G_1 by one, and may apply induction to find a coloring with all reduced vertices monochromatic.

Step 9. Decontract unbridged reduced vertices.

Step 10. Decontract paths of reduced vertices. After this step, each component of G_2 is a Θ-graph: a pair of vertices with three disjoint paths between them. Also, all degree-two vertices of these theta-graphs are reduced vertices.

Step 11. Reintroduce cycles and near-cycles of reduced vertices. After this step, each color class of the coloring has components that are either paths or cycles. Furthermore, each reduced vertex that we have reintroduced in this step is monochromatic, so all reduced vertices of G_1 are still monochromatic.

Step 12. Color the Θ-components. Let one color class be a cycle and the other a path; the reduced vertices whose edges are colored in this step are monochromatic.

We now have G' (the full G' constructed in Step 1) edge-colored so that each reduced vertex is monochromatic, and each color class has components that are either paths or cycles.

Step 13. Reintroduce high-degree vertices. Put back all vertices and edges removed in Step 1. Each edge being replaced in this step is given color opposite to the color of the reduced vertex that is its endpoint. This creates a red star and a blue star for each high-degree vertex reintroduced; these stars do not connect to any other components of their color. We now have G colored so that each color class has components that are either paths, cycles, or stars. No two cycles of opposite colors intersect, as such an intersection must take place at a vertex of degree at least four, and no cycle goes through a high-degree vertex.

Step 14. Lay out as rectangles.

□

Theorem 10. *Every 3-hilly graph G is a collinear RVG, and a collinear layout of G can be constructed in linear time.*

The proof of this theorem proceeds as in the proof of Theorem 9, preceding Step 14 with some extra graph manipulation so as to produce a collinear RVG rather than a weak one. This is done by adjusting the two-coloring to satisfy the hypothesis of Lemma 8; we omit the details. The following theorem is proved similarly; the cases where collinearity is used in Theorem 10 cannot occur in a 4-hilly* graph.

Theorem 11. *Every 4-hilly* graph is a noncollinear RVG, and a noncollinear layout of G can be constructed in linear time.*

Note that the technique used in above three theorems can be pushed even farther, to show that RVGs include graphs consisting of a few small clusters of high-degree vertices in a sea of low-degree vertices:

Theorem 12. *Let G be a graph, and H be the graph induced on the high-degree vertices of G. If H is maximum-degree three, and $G \setminus E(H)$ is 2-hilly, 3-hilly, or 4-hilly*, then G is a weak RVG, a collinear RVG, or a noncollinear RVG, respectively. Furthermore, a layout of such a graph can be constructed in linear time.*

We can also show that the results of this section cannot be strengthened to include 1-hilly graphs; there is a 1-hilly graph that is not a weak RVG.

7 Maximum Degree Four

In this section, we show that every maximum-degree 4 graph is a weak RVG. As with the k-hilly proofs, there are two distinct components to the proof: one graph-theoretic, and the other geometric. Here, however, because the graph-theoretic part is less detailed, we present that part before establishing the necessary geometric result.

Theorem 13. *Every maximum-degree 4 graph G is a weak RVG, and a weak layout of G can be computed in linear time.*

Proof. Start by augmenting the graph G with extra edges and vertices so that it becomes a 4-regular graph G'. Next, find an Euler circuit in G'; this is possible, as every vertex of G' has even degree. Alternately color the edges of the circuit red and blue. The Euler circuit will pass through each vertex twice, so each vertex will be incident on two red edges and two blue edges. This means that each color class is a 2-factor, or, more simply, a collection of cycles.

We prove below (as Theorem 14) that any such graph (one that can be decomposed into two collections of cycles) is a weak RVG, with layout taking linear time. Lay out G' as a weak RVG, and then remove any rectangles corresponding to vertices of G' that are not vertices of G. The extra edges of G' and the possible extra visibilities introduced when removing rectangles are inconsequential under the definition of weak RVGs, so the resulting layout is a weak layout of G. □

Theorem 14. *Let G be a graph that can be decomposed (edge-colored) into two graphs whose connected components are either caterpillars or cycles. The graph G is a weak RVG, and a weak layout of G can be computed in linear time.*

Proof. This is Lemma 7 with the restriction that two cycles intersect in at most one vertex removed. That restriction was to make the rightmost or topmost rectangle in a cycle well-defined. Removing it means that we will have to arrange things so that we can safely choose such a rightmost or topmost rectangle.

We assume that the given graph G is connected. Let us label the connected components of the blue subgraph $C_1, C_2, \ldots, C_k$ (C stands for column) and of the red subgraph $R_1, R_2, \ldots, R_l$ (R stands for row). Unlike in previous proofs, we will be careful about which order the rows and columns are placed in.

Whenever we lay out a row, we place it immediately below all rows that have already been laid out, and above all rows yet to be laid out. Similarly, we lay out columns to the left of all columns already laid out. The only real difficulty is getting started; i.e. choosing the rightmost column and topmost row, and showing how to lay out these components. We analyze several cases.

Case 1. There is a component (wlog a column) C_i that is a caterpillar. We let this column be the rightmost, and choose any row R_j that intersects C_i as the topmost row (such a row exists by connectivity of G). Lay out C_i as usual, and if R_j is a caterpillar, lay it out as usual also. If R_j is a cycle, then it has a rightmost vertex v (i.e. one whose horizontal interval's right endpoint is rightmost) in $R_j \cap C_i$, and we can lay it out as we did in Lemma 7, adjusting C_i if necessary.

All further cases assume that all components are cycles.

Case 2. There are two components C_i and R_j whose intersection contains exactly one vertex v. Let this row and column be the topmost and rightmost. Let the rectangle for v be the entire intersection of the horizontal and vertical intervals for C_i and R_j. Both $C_i \setminus v$ and $R_j \setminus v$ are paths and can be laid out as such inside their respective intervals; no visiblities are duplicated and therefore no adjustments need to be made.

Case 3. There are two components C_i and R_j whose intersection contains at least two vertices v and w, and furthermore, v and w are adjacent in one of these components (wlog in C_i). First, let v be placed so that its vertical interval is the vertical interval for R_j, and its horizontal interval is a small interval at the right end of the interval for C_i. We lay out $R_j \setminus v$ as a path as usual.

Let I be the horizontal interval for C_i with the interval for v removed. We lay out $C_i \setminus v$ as a path in I, with w on the left. The visibility between v and w will be horizontal. We must also adjust the layout depending on whether the other vertex u adjacent to v in C_i is in $C_i \cap R_j$: if it is, the interval for u should be contained in I (so that v sees u horizontally), and if not, this interval should extend slightly to the right of I so that u sees v vertically.

Case 4. No other case above holds. Choose a component (wlog a row R_b) whose removal will not disconnect the graph. The row R_b will be the *bottom* row and will be processed last. Let z be a vertex of R_b, and C_i the column containing z. Let y be a vertex adjacent to z in the cycle C_i, and R_j be the row containing z. The rows R_j and R_b are not the same, or else we would be in Case 3.

The components C_i and R_j will be the rightmost and topmost, and we choose $v = y$ and $w = z$ and lay out C_i and R_j as in Case 3, where the the path $C_i \setminus v$ stays strictly within the interval I. An example of this construction is shown in Figure 15. We have now laid out these components, but have not yet established the visibility between v and u, or between v and $w = z$. We will adjust v and lay out R_b at the end of the entire construction so as to establish these visibilities.

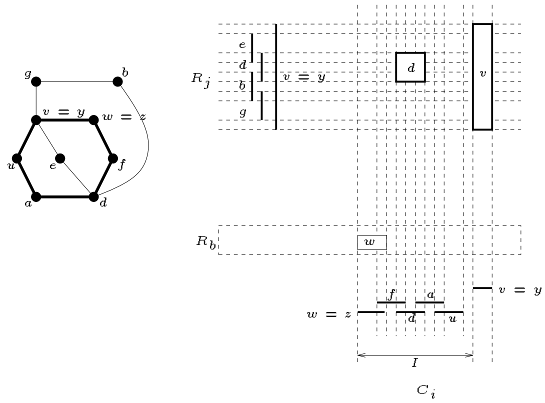

Fig. 15. Case 4 in the proof of Theorem 14.

In any case, we have laid out a topmost and rightmost component. The rest is straightforward:

Until all components have been laid out (except R_b if we were in Case 4), choose a component that is not laid out and that intersects a component that has already been laid out. (Such a component always exists, by connectivity of G or of $G \setminus R_b$.) Lay out this newly chosen component as follows:

Wlog, the component is a row R_j. If R_j is a caterpillar, lay it out as usual. Otherwise, let C_i be the rightmost column that has nonempty intersection with

R_j, and v be the rightmost vertex of this column in their intersection. Let v have the vertical interval corresponding to the entire interval for R_j, and lay out the path $R_j \setminus v$ inside the interval for v. We may have duplicated visibilities already present in C_i; if this is the case then we remove them as in Lemma 7 or by a special-case adjustment, as shown in Figure 16.

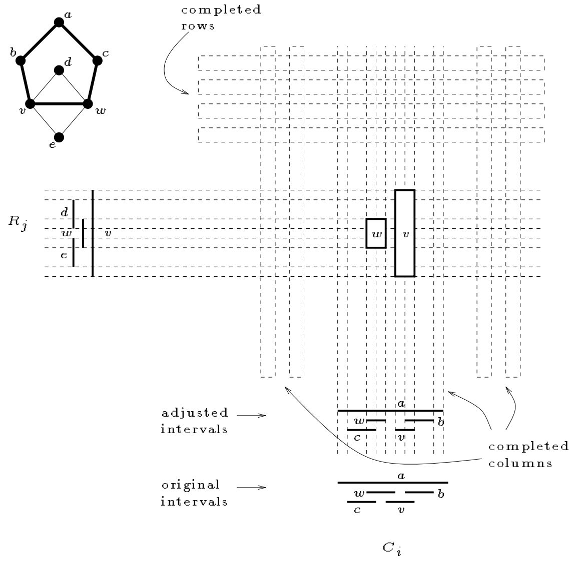

Fig. 16. The general step in Theorem 14.

This ends the general step. If we were in case 4, we have to lay out R_b and adjust the layout to get visibilities between the original v and u and between the original v and w; the details of this are omitted. □

8 Conclusion

We have shown that several classes of graphs are RVGs, including 4-trees, caterpillar arboricity two, maximum degree 4, and 2-hilly. We examined several subclasses of these classes, and established that some of them are collinear or noncollinear RVGs. In each case we have linear layout algorithms. Of particular interest is the class of graphs with maximum degree 3, which can be laid out with all rectangle sizes prespecified.

This abstract summarizes two manuscripts [2, 10]. In subsequent work, Shermer [11] has shown that recognizing RVGs is NP-complete, as is the problem of recognizing graphs with caterpillar arboricity two.

The question of where maximum-degree 4 and maximum-degree 5 graphs truly fall is still unsettled. It may be the case that maximum-degree 4 graphs are collinear or even noncollinear RVGs. On the other hand, Dean and Hutchinson [4] have shown that $K_{5,5}$ is not a collinear RVG, but it is a weak RVG ($K_{5,5}$ plus any edge is a collinear RVG). So maximum-degree 5 graphs either include some graphs that are not RVGs of any sort, or they are a subclass of weak RVGs.

Another question that is open is whether or not every graph with arboricity two is an RVG of some sort. As every forest (graph with arboricity one) is a BVG, one might suspect that this could be true.

In closing, we would like to acknowledge Peter Eades for suggesting some of the motivational ideas, Diane Souvaine and Alan Taylor for helpful discussions, and Stan Wagon for assisting with the figures.

References

1. T. Biedl. personal communication, 1995.
2. P. Bose, A. Dean, J. Hutchinson, and T. Shermer. On rectangle visibility graphs I. k-trees and caterpillar forests. manuscript, 1996.
3. A. M. Dean and J. P. Hutchinson. Combinatorial representations of visibility graphs. preprint, 1996.
4. A. M. Dean and J. P. Hutchinson. Rectangle-visibility representations of bipartite graphs. *Discrete Applied Mathematics*, 1996, to appear.
5. P. Duchet, Y. Hamidoune, M. Las Vergnas, and H. Meyniel. Representing a planar graph by vertical lines joining different levels. *Discrete Mathematics*, 46:319–321, 1983.
6. M. R. Garey, D. S. Johnson, and H. C. So. An application of graph coloring to printed circuit testing. *IEEE Transactions on Circuits and Systems*, CAS-23:591–599, 1976.
7. P. Horak and L. Niepel. A short proof of a linear arboricity theorem for cubic graphs. *Acta Math. Univ. Comenian.*, XL–XLI:255–277, 1982.
8. J. P. Hutchinson, T. Shermer, and A. Vince. On representations of some thickness-two graphs (extended abstract). In F. Brandenburg, editor, *Proc. of Workshop on Graph Drawing*, volume 1027 of *Lecture Notes in Computer Science*. Springer-Verlag, 1995.
9. F. Luccio, S. Mazzone, and C. K. Wong. A note on visibility graphs. *Discrete Mathematics*, 64:209–219, 1987.
10. T. C. Shermer. On rectangle visibility graphs II. k-hilly and maximum-degree 4. manuscript, 1996.
11. T. C. Shermer. On rectangle visibility graphs III. external visibility and complexity. manuscript, 1996.
12. R. Tamassia and I.G. Tollis. A unified approach to visibility representations of planar graphs. *Discrete and Computational Geometry*, 1:321–341, 1986.
13. S. K. Wismath. Characterizing bar line-of-sight graphs. In *Proc. 1st Symp. Comp. Geom.*, pages 147–152. ACM, 1985.
14. S. K. Wismath. *Bar-Representable Visibility Graphs and a Related Network Flow Problem.* PhD thesis, Department of Computer Science, University of British Columbia, 1989.