Recognizing some subclasses of vertex intersection graphs of 0-bend paths in a grid[⋆]

Steven Chaplick[1], Elad Cohen[2], and Juraj Stacho[2]

[1] Department of Computer Science, University of Toronto, 10 Kings College Road, Toronto, Ontario M5S 3G4, Canada, (chaplick@cs.toronto.edu)
[2] Caesarea Rothschild Institute, University of Haifa, Mt. Carmel, Haifa, Israel 31905 (eladdc@gmail.com and stacho@cs.toronto.edu)

Abstract. We investigate graphs that can be represented as vertex intersections of horizontal and vertical paths in a grid, known as B_0-VPG graphs. Recognizing these graphs is an NP-hard problem. In light of this, we focus on their subclasses. In the paper, we describe polynomial time algorithms for recognizing chordal B_0-VPG graphs, and for recognizing B_0-VPG graphs that have a representation on a grid with 2 rows.

1 Introduction

A *VPG representation*[†] of a graph G is a collection of paths of the two-dimensional grid where the paths represent the vertices of G in such a way that two vertices of G are adjacent if and only if the corresponding paths share at least one vertex. We focus on a special subclass of VPG representations.

A *B_0-VPG representation* of G is a VPG representation in which all paths in the collection have no bends. In other words, it is a representation of G by the intersections of orthogonal segments of the plane. Here, we emphasize the grid-based definition in order to focus on some properties of the underlying grid (e.g. size). A graph is a *B_0-VPG graph* if it has a B_0-VPG representation.

Intersection representations of paths on grids arise naturally in the context of circuit layout problems and layout optimization [18] where a layout is modeled as paths (wires) on a grid. Often one seeks to minimize the number of times a wire is bent [3, 17] in order to minimize the cost or difficulty of production. Other times layouts may consist of several layers where the wires on each layer are not allowed to intersect. This is naturally modeled as the colouring problem on the corresponding intersection graph.

VPG graphs were defined in [1, 2] where, in particular, the subclasses with bounded number of bends are studied. These classes are shown to have many connections to other, more traditional graph classes such as interval graphs, planar graphs, string graphs, segments graphs, circle graphs and circular-arc graphs. Unfortunately, due to these connections, many natural problems for VPG graphs are hard. For instance, colouring is NP-hard even for B_0-VPG graphs, and recognition is NP-hard for both VPG and B_0-VPG graphs (for more details about these and related results, see [2, 13, 14]).

[⋆] the authors wish to thank Krishnam Raju Jampani, Therese Biedl, and Martin Charles Golumbic for fruitful discussions in the early stages of this work

[†] representation by Vertex intersection of Paths in a Grid

Thus, in the quest for polynomial algorithms, we need to restrict our attention further to specific cases with (potentially) useful structure. In this respect, in [8], certain subclasses of B_0-VPG graphs have been characterized and shown to admit polynomial time recognition; namely split, chordal claw-free, and chordal bull-free B_0-VPG graphs are discussed in [8].

In this paper, we continue this line of research by further investigating two other subclasses of B_0-VPG graphs. In particular, we describe a polynomial time algorithm for recognizing chordal B_0-VPG graphs, and a polynomial time algorithm for recognizing 2-row B_0-VPG graphs, i.e., B_0-VPG graphs that can be represented on a grid with just 2 rows (and arbitrary number of columns). Note that the former generalizes [8] and one can easily verify that the underlying grid graph induced by the paths of a chordal B_0-VPG graph is, in fact, a tree.

Studying B_0-VPG representation of chordal graphs is a natural choice as they are precisely the intersection graphs of subtrees of a tree, and can be also seen as the intersection graphs of leaf generated subtrees of a complete binary host tree [11, 16] which by [10] has a near optimal embedding on a grid. Moreover, the colouring problem can be solved in linear time on chordal graphs (see [7]). Similarly, the choice of 2-row representation of B_0-VPG graphs is a natural one since when considering embeddings of graphs in grids, one objective is to utilize as little space as possible; in this context, 2-row representations constitute the smallest non-trivial case one can study and one that has not been considered before this work. In the conclusion of the paper, we discuss the complexity of the colouring problem on such representations (with bounded number of rows).

Both our recognition algorithms are based on essentially the same idea which follows from the realization that the rows and columns of the grid induce interval representations. That is, a graph G is a B_0-VPG graph if there is a partition of its vertex set such that each class of the partition induces in G an interval graph and the connections between the classes follow "certain" structure. There are two specific problems related to this approach: we need to find the partition (it is easy to test if each class is an interval graph), and we need an efficient way to construct a representation of the connection graph of the classes.

We deal with these questions differently in each case. In the case of chordal B_0-VPG graphs, the classes are based on an equivalence relation on the vertices. The connection graph turns out to be a tree, and we describe an algorithm to find a layout of this tree which yields a representation of G. In the case of 2-row B_0-VPG graphs, the classes are found by splitting *bisimplicial* vertices (vertices whose neighbourhood induces two cliques with no edges between them). The connection graph of the classes is a planar graph that can be drawn on two layers without crossings, and this drawing satisfies additional conditions.

The two cases are discussed separately in the following sections. We mostly follow the definitions and notation from [7] and [19]. In the interest of brevity, we shall assume some familiarity with *chordal graphs*, *interval graphs*, *clique paths* (linear orderings of maximal cliques [6]), *planar drawings*, and *PQ-trees* [4].

A vertex of G is *horizontal*, resp. *vertical* in a B_0-VPG representation of G if it is represented by a horizontal, resp. vertical path.

2 Chordal B_0-VPG graphs

In this section, we describe a polynomial time algorithm for recognizing chordal B_0-VPG graph. First, we recall the following lemma from [8].

(2.1) Diamond rule. *Let G be the graph with $V(G) = \{u, v, x, y\}$ and $E(G) = \{uv, ux, uy, vx, vy\}$. Then in every B_0-VPG representation of G, the two paths representing u and v use a common horizontal or a common vertical line.*

This inspires the following definition.
Let G be a graph. The binary relation $\sim_0$ on $V(G)$ is defined as follows:

$u \sim_0 v \iff uv \in E(G)$ and $\exists x, y \in N(u) \cap N(v)$ with $xy \notin E(G)$

In other words, u and v are related by $\sim_0$ if they form the diagonal of some diamond. Let $\sim$ denote the transitive closure of $\sim_0$.

(2.2) *Let G be a chordal graph, S be an equivalence class of $\sim$, and K be a connected component of $G - S$. Then $(N(S) \cap K) \cup (N(K) \cap S)$ is a clique of G.*

Proof (Sketch). If there are no edges between K and S we are done. Otherwise, suppose that there is $x \in N(K) \cap S$ and $y \in N(S) \cap K$ such that $xy \notin E(G)$. Since $x \in N(K)$, there exists $y' \in K$ with $xy' \in E(G)$, and since $y \in N(S)$, there is $x' \in S$ where $x'y \in E(G)$. Choose x, y, x', y' so that $d_S(x, x') + d_K(y, y')$ is minimized where $d_S(x, x')$ is the distance between x and x' in $G[S]$, and $d_K(y, y')$ is the distance between y and y' in $G[K]$. By the minimality of the choice and chordality of G, we conclude that $xx', yy' \in E(G)$ and $x'y' \in E(G)$. This shows that $x' \sim y'$ which is impossible, since $x' \in S$, $y' \notin S$ and S is an equivalence class of $\sim$. So, there is no such x, y and the rest of the claim follows easily. $\square$

The *clique-class intersection graph* (see Fig. 1) of G is the bipartite graph whose vertices are the maximal cliques of G and the equivalence classes of $\sim$, where a clique Q is adjacent to a class S just if they share at least one vertex.

(2.3) *If G is a connected chordal B_0-VPG graph, then the clique-class intersection graph of G is a tree.*

(Note that the clique-class intersection graph of G is precisely the block-cutpoint tree of the graph we obtain from G by contracting all equivalence classes of $\sim$.)

The proof of the following claim follows directly from (2.2).

(2.4) *Let G be a chordal B_0-VPG graph, and S be an equivalence class of $\sim$. Then the closed neighbourhood $N[S]$ of S induces in G an interval graph.*

Consider a B_0-VPG representation of G, and let Q be a maximal clique of G. Let I denote the intersection of the paths representing the vertices in Q. By the Helly property [2], the set I is non-empty. Let S be an equivalence class of $\sim$ such that $Q \cap S \neq \emptyset$. We say that Q is an *end* of S in this representation, if the vertices of $S \setminus Q$ are represented by paths that are either all to the right, or all to the left, or all above, or all below all points in I.

We say that Q is a *forced end* of S if Q is an end of S in every B_0-VPG representation of G. We say that Q is a *forced midpoint* of S if there is no B_0-VPG representation of G in which Q is an end of S.

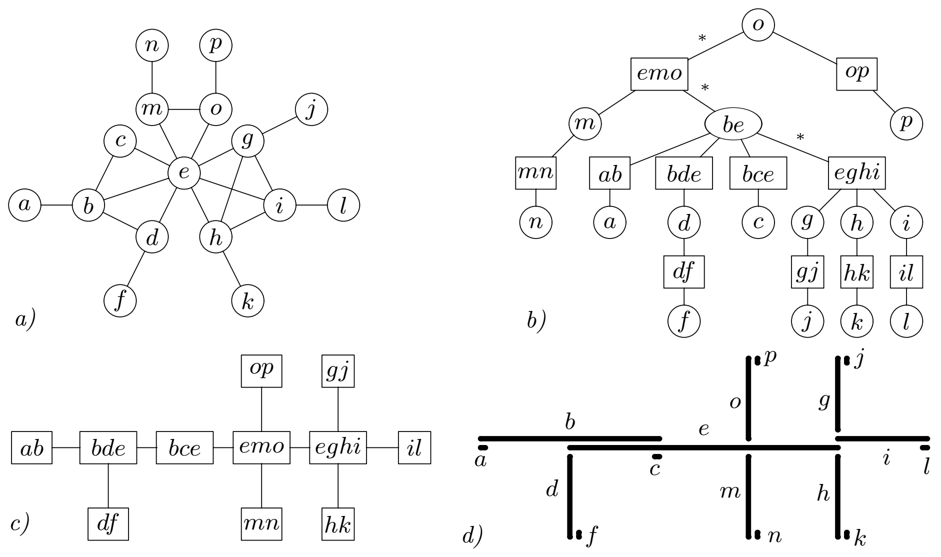

Fig. 1. *a)* Example chordal graph G, *b)* the clique-class intersection graph H of G (edges with * are those marked by the algorithm), *c)* the graph T=the union of the clique paths P_S after identifying cliques, *d)* the corresponding B_0-VPG representation

The following two claims explain the role of forced ends and midpoints in chordal B_0-VPG graphs. They are simple consequences of (2.1), (2.2), and (2.4).

(2.5) *Let G be a chordal B_0-VPG graph. Let Q be a maximal clique of G, and let $S_1, \ldots, S_t$ be all the equivalence classes of $\sim$ that have non-empty intersection with Q. Let r be the number of indices $i \in \{1 \ldots t\}$ for which Q is a forced midpoint of S_i. Let s be the number of indices $i \in \{1 \ldots t\}$ with $Q \supseteq S_i$.*

Then $r + t - s \leq 4$. Moreover, if there exists $i \in \{1 \ldots t\}$ such that Q is neither a forced end of S_i nor a forced midpoint of S_i, then $r + t - s \leq 3$.

(2.6) *Let G be a chordal B_0-VPG graph. Let S be an equivalence class of $\sim$, and let $Q_1, \ldots, Q_t$ be the maximal cliques of G with non-empty intersection with S.*

Construct a graph G' starting from $G[N[S]]$ as follows: for each $i \in \{1 \ldots t\}$ such that Q_i is a forced end of S, add a new vertex u_i and make it adjacent to each vertex in $Q_i \setminus S$. Then G' is an interval graph.

Moreover, if for $i \in \{1 \ldots t\}$ adding to G' a vertex u_i adjacent to each vertex in $Q_i \setminus S$ results in a non-interval graph, then Q_i is a forced midpoint of S.

2.1 Algorithm

Now, we are ready to describe our algorithm for recognition of chordal B_0-VPG graphs. Let G be a graph given as input. We may assume that G is connected. Otherwise, we obtain a representation of G by finding a representation for each of its connected components, and by putting the representations side-by-side.

First, we check if G is chordal (see [7]). If not, we reject G. Otherwise, we compute the maximal cliques of G and the equivalence classes of $\sim$. We test if the closed neighbourhood of each equivalence class induces in G an interval graph. If not, we reject G based on (2.4). Otherwise, we construct the clique-class intersection graph H. By (2.3), the graph H is in fact a tree. The algorithm uses dynamic programming on H to find out which cliques of G are forced ends or forced midpoints of equivalence classes. To test this, we use (2.5) and (2.6).

We start by rooting H at an arbitrary node. The nodes of H are processed bottom-up, processing a node only after all its descendants are processed. We mark some edges of H in this process; initially, no edges are marked, and once an edge becomes marked, it remains marked. The meaning of a marked edge e between a clique Q and class S is the following. If Q is the parent of S, and e becomes marked when processing S, then Q is a forced midpoint of S (otherwise, Q is not is not a forced midpoint of S). Similarly, if Q is a child of S, and the edge e becomes marked when processing Q, then Q is a forced end of S.

When processing a clique Q, we count the number of children S such that the edge between Q and S is marked. Thus, Q is a forced midpoint in each such S and is not a forced midpoint in all other children. If Q is the root of H, we apply the test from (2.5). If this fails, we reject G. If Q has a parent S^*, we test (2.5) assuming that Q is not a forced midpoint of S^*. If this fails, we reject G. If only the second part of (2.5) fails, then Q is necessarily a forced end of S^*, and we mark the edge between Q and S^*. Otherwise, we do not mark the edge.

Similarly, we process each class S using (2.6). First, we look at the marked children Q of S; each such Q is a forced end of S, and we shall assume that all other children are not. If S is the root of H, we just perform the first test from (2.6), and if it fails, we reject G. Otherwise, if S has a parent Q^*, we use (2.6) to determine whether or not Q^* is a forced midpoint of S. If so, we mark the edge between S and Q^*. If not, we conclude that Q^* is not a forced midpoint of S (by providing a representation), and thus we do not mark the edge.

It remains to explain how we obtain a representation of G if this process finishes without rejecting G. For each class S, we assign to S the interval representation of G' guaranteed by (2.6); in this representation every forced end of S is necessarily an end of S. If the parent of S is not a forced midpoint, we assign to S the representation from the second part of (2.6); in this representation, the parent of S is an end of S. From this representation, we remove the vertices that are not in G (the vertices u_i added in the process of creating G'), and consider the resulting representation as an equivalent clique path that we denote by P_S. Observe that the cliques on this path are maximal cliques in G.

Now, consider the graph obtained by taking the disjoint union of the above clique paths; each vertex corresponds to some maximal clique of G, and each connected component is the path P_S for some S. In this graph, for each clique Q, we find all vertices that correspond to Q, and identify them to a single node. This results in a graph T. We observe that T is a tree, since H is a tree. In fact, T is a clique tree, since the paths P_S are clique paths. By the choice of the paths using (2.6) and since each clique Q satisfies (2.5), the tree T has maximum degree four, and hence, can be drawn in the plane so that the edges are represented by horizontal or vertical segments. In fact (by appropriately permuting neighbors), we can draw T so that each path P_S is horizontal or vertical in the drawing. Since each vertex of G belongs to exactly one equivalence class S, we conclude that it appears only in cliques on the path P_S, and we represent it by a path connecting all such cliques on P_S. This yields a B_0-VPG representation of G.

That concludes the description of our algorithm. We now briefly analyze its running time. Testing for chordality takes linear time (see [7]), so does computing all maximal cliques (there is $O(n)$ of them). Finding all diamonds and thus computing the equivalence classes takes $O(nm)$ time. Having done that, the construction of the clique-class intersection graph H takes linear time. Finally, since H has $O(n)$ nodes, the dynamic programming takes $O(nm)$ time.

Thus, the recognition algorithm runs in $O(nm)$ time.

3 2-row B_0-VPG graphs

A *2-row B_0-VPG representation* of G is a B_0-VPG representation where the underlying grid has two rows (and arbitrary number of columns). We call the two rows *layers* and distinguish the *top* and the *bottom* layer. For simplicity, any path of the representation that is a single grid-point is considered to be horizontal. Thus, a vertical path always consists of exactly two points of the grid (in the same column), one on the top and one on the bottom layer.

A graph is a *2-row B_0-VPG graph* if it has a 2-row B_0-VPG representation. In this section, we describe a polynomial time algorithm for recognizing 2-row B_0-VPG graphs. It suffices to focus on connected graphs. Also, it suffices to consider graphs with no true twins, where u, v are *true twins* if $N[u] = N[v]$. Clearly, if u, v are true twins in G, then a representation of G can be obtained from a representation of $G - v$ by assigning to v the path corresponding to u.

A vertex v of G is *bisimplicial* if the neighbourhood of v in G induces a disjoint union of two non-empty cliques. In other words, the set $N(v)$ induces in G a graph consisting of two non-empty cliques with no edges between them. (See Fig. 2 for an illustration of this and subsequent notions.)

A B_0-VPG representation of G is *proper* if for every path of the representation, each of its endpoints belongs to at least one other path. It suffices to consider only proper representations. We note the following properties.

(3.1) *If v is a bisimplicial vertex of G, then in every B_0-VPG representation of G, some edge of the underlying grid belongs only to the path representing v.*

(3.2) *In each proper 2-row B_0-VPG representation of a graph G with no true twins, every vertical vertex is a bisimplicial vertex of G.*

Proof. If v is a vertical vertex, then, since the representation is proper, the vertical path P representing v intersects both a horizontal path on the top layer and a horizontal path on the bottom layer of the representation. Thus the horizontal paths intersecting P on the top layer and the horizontal paths intersecting P at the bottom layer yield the two cliques in $N(v)$ as required. □

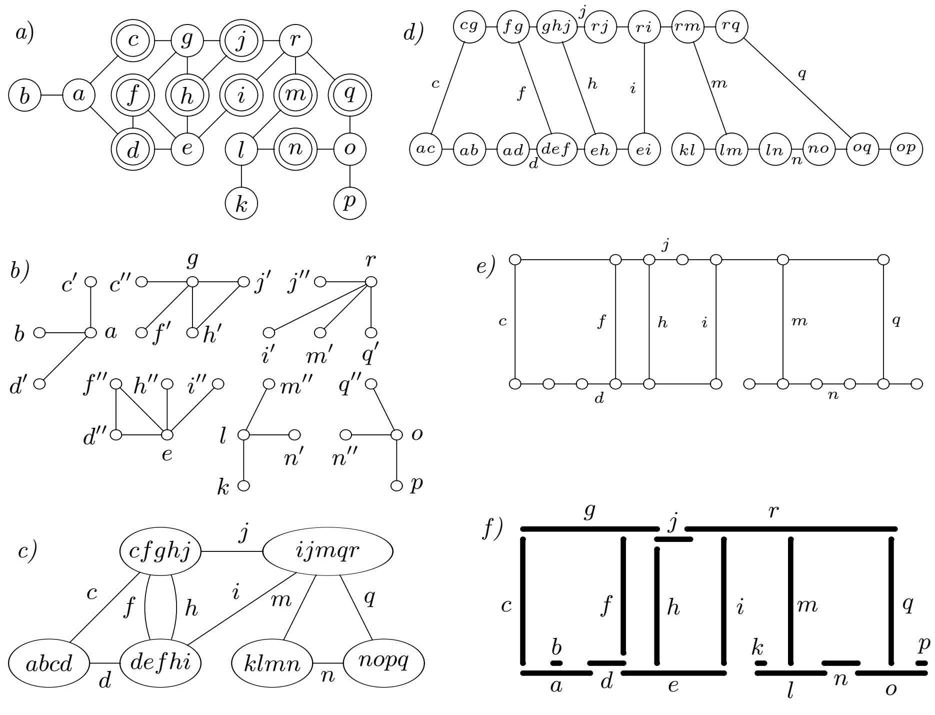

Fig. 2. *a)* Example graph G (bisimplicial vertices shown in double circles), *b)* the bisplitting of G, *c)* the bicontraction H of G, *d)* the graph H'= substituting clique paths to H, *e)* orthogonal drawing of H', *f)* the corresponding 2-row B_0-VPG representation

The converse of (3.2) is false, i.e., not all bisimplicial vertices are necessarily represented by vertical paths. To find out which are, we use two auxiliary graphs.

Let G be a graph. The <u>*bisplitting*</u> (see Fig. 2) of G is the graph obtained from G as follows. We consider each bisimplicial vertex v of G and let Q_1, Q_2 denote the two cliques induced by $N(v)$. We remove v, add new vertices v', v'', make v' adjacent to each vertex in Q_1, and make v'' adjacent to each vertex in Q_2.

(3.3) *If G is a 2-row B_0-VPG graph with no true twins, then the bisplitting of G is an interval graph.*

Proof. Consider a proper 2-row B_0-VPG representation of G. By (3.1), each bisimplicial vertex v of G has an edge e of the underlying grid that only belongs to v's path P. We remove e from P to obtain two subpaths P_1 and P_2. Since the representation is proper, the paths representing the vertices of one of the two cliques in $N(v)$ intersect P_1, while the paths for the other clique intersect P_2. We represent v' by P_1 and v'' by P_2. Now, by (3.2), no vertical paths remain in the representation. Thus the result is an interval representation. $\square$

The <u>*bicontraction*</u> (see Fig. 2) of G is the multigraph H described as follows. The vertex set of H is the set of connected components of the bisplitting of G. There is an edge e with label v between connected components C', C'' if and only if v is a bisimplicial vertex of G that was "split" into v' and v'' where $v' \in C'$ and $v'' \in C''$. We allow possibly $C' = C''$ in which case e is a loop.

For further use, we denote by $Q_e^{C'}$ the closed neighbourhood of v' in C' and denote by $Q_e^{C''}$ the closed neighbourhood of v'' in C''. Note that, by the definition of bisplitting, $Q_e^{C'}$ and $Q_e^{C''}$ are maximal cliques in C' and C'', respectively.

(3.4) *If G is a 2-row B_0-VPG graph, then the bicontraction of G contains no loops, and no two parallel edges have labels that are adjacent in G.*

3.1 LL-drawings

A *planar LL-drawing* of a multigraph H is a planar embedding of H on two layers. In other words, the vertices are placed on two horizontal lines such that every edge is drawn (without crossing another edge) either between consecutive vertices on a layer or between the two layers, and there are no parallel edges between vertices on the same layer. A linear time algorithm for finding planar LL-drawings is discussed in [5]. An edge of a planar LL-drawing is *horizontal* (respectively *vertical*) if its endpoints are on one (respectively two) layers.

A vertical edge e of a planar LL-drawing $\mathcal{E}$ splits the drawing into two areas, the area L_e to left of e, and the area R_e to the right e. For any edge e', we write $e' \prec_{\mathcal{E}} e$ (respectively $e' \succ_{\mathcal{E}} e$) if the interior of e' belongs to L_e (respectively R_e). For all other pairs of edges, we derive $\prec_{\mathcal{E}}$ by transitivity. Since there are no crossings, the resulting relation $\prec_{\mathcal{E}}$ is a partial order on the edges of $\mathcal{E}$; it is a total order on the vertical edges of $\mathcal{E}$. We refer to $\prec_{\mathcal{E}}$ as the *$\mathcal{E}$-order* of the edges.

Using the proof of (3.3), we now *associate* to each 2-row B_0-VPG representation of G with no true twins a planar LL-drawing of the bicontraction of G.

We proceed as follows. Starting with a 2-row B_0-VPG representation of G, we denote by H_0 the subgraph of the underlying grid obtained by taking the union of all paths of the representation. Using the proof of (3.3), we obtain a representation of the bisplitting G' of G. During this process, a set F of edges is removed from the paths of the representation. By definition, each edge in F is also an edge of H_0. Further, by the construction, each connected component K of $H_0 - F$ corresponds to some connected component C of G' in that K is a horizontal path that includes all paths representing the vertices of C.

Now, label each edge $e \in F$ with the bisimplicial vertex of G from whose path we removed e. After that, consider the multigraph H obtained by contracting the edges of H_0 that are not in F (while keeping parallel edges). Notice that each connected component of G' shrinks to one point and F is the edge-set of H. Thus, we conclude that H is the bicontraction of G.

Finally, note that H_0 itself (as a subgraph of the underlying grid) is a planar LL-drawing, and we obtained H by contracting horizontal edges in this drawing. Thus, the result is a planar LL-drawing of H, the bicontraction of G, and we associate it to the 2-row B_0-VPG representation of G we started with.

Notice that this associated LL-drawing has some special properties. They are summarized in the following definition. Again, let H be the bicontraction of G, and let $\mathcal{E}$ be a planar LL-drawing of H. Let C be a vertex of H, and let $e_1, \ldots, e_t$ be the edges incident to C in H such that $e_1 \prec_{\mathcal{E}} \ldots \prec_{\mathcal{E}} e_t$.

We say that C is good in $\mathcal{E}$ if there exists a clique path $\mathcal{P}$ of C such that

(L1) for all $i < j$, if $Q^C_{e_i} \neq Q^C_{e_j}$, then $Q^C_{e_i}$ appears before $Q^C_{e_j}$ on $\mathcal{P}$,

(L2) for all $i < j$, if $Q^C_{e_i} = Q^C_{e_j}$, then at least one of the following holds:
  (L2a) $i = 1$, $j = 2$, and e_1 is a horizontal edge,
  (L2b) $i = (t-1)$, $j = t$, and e_t is a horizontal edge,
  (L2c) $i = 1$, $j = t = 3$, and $Q^C_{e_1} = Q^C_{e_2} = Q^C_{e_3}$,

(L3) if e_1 is a horizontal edge, then $Q^C_{e_1}$ is the first clique on $\mathcal{P}$, and
  if e_t is a horizontal edge and $t \geq 2$, then $Q^C_{e_t}$ is the last clique on $\mathcal{P}$.

We say that $\mathcal{E}$ is a good LL-drawing if every vertex of H is good in $\mathcal{E}$.

It is not difficult to see that the associated planar LL-drawing of a 2-row B_0-VPG representation of G is a good LL-drawing. Namely, for each vertex C, we use the clique path $\mathcal{P}$ corresponding to the interval representation of C induced by the representation of G. Then $\mathcal{P}$ (or its reverse†) satisfies the above conditions. It turns out that the converse is also true yielding the following characterization.

(3.5) *A graph G has a 2-row B_0-VPG representation if and only if there exists a good LL-drawing of the bicontraction of G.*

Proof. The forward direction is discussed above the claim. For the backward direction, we consider a good LL-drawing of the bicontraction H of G. In the drawing, we replace each vertex C by the clique path $\mathcal{P}$ of C satisfying (L1)-(L3). We arrange the vertices of $\mathcal{P}$ from left to right in the order given† by $\mathcal{P}$. For each edge e incident to C, we reattach e from C to Q^C_e (which is one of the vertices on $\mathcal{P}$). From (L1)-(L3) we conclude that the result of this process is a planar LL-drawing of a graph H'. In particular, the vertices of H' correspond precisely to the maximal cliques of G. By (possibly) adding gaps between consecutive vertices on layers, we modify the drawing so that every vertical edge of the drawing is drawn as a vertical segment, and vertices are on integer coordinates. Finally, we assign to each vertex v of G the path in the drawing connecting all cliques of G that contain v. This yields a 2-row B_0-VPG representation of G. $\square$

3.2 PQ-trees

To find a good LL-drawing of the bicontraction of G, we use the well-known concept of a PQ-tree [4]. We briefly review some key properties of such trees.

A PQ-tree is a *rooted ordered* tree (children of each node are totally ordered) where each internal node is either a P-node or a Q-node, and whose *leaf order* is defined as the ordering of leaves in the traversal that visits the children of each node in the given total order. Two PQ-trees are *equivalent* if one can be obtained from the other by possibly permuting the children of some P-nodes and reversing the order of children of some Q-nodes. A permutation π of a set is *consistent* with a PQ-tree if there exists an equivalent PQ-tree whose leaf order is π.

If T, T' are PQ-trees with the same leaf-set, then the *intersection* of T and T' is the PQ-tree whose consistent permutations are precisely those that are consistent with both T and T'. If no such permutations exist, the intersection is the *null tree*. The intersection takes linear time to construct (for instance, see [9]).

† if C belongs to only one horizontal edge, we may need to reverse $\mathcal{P}$

3.3 Algorithm

Now, we are ready to describe our recognition algorithm for 2-row B_0-VPG graphs. By (3.5), it suffices for the algorithm to find a good LL-drawing of the bicontraction of the given graph G. This, up to minor technical details, boils down to the following problem: given a multigraph H with a collection of PQ-trees $\{T_v\}_{v \in V(H)}$, find a planar LL-drawing of H such that the clockwise ordering of edges around each $v \in V(H)$ is consistent with T_v. Our algorithm follows this idea. We assign to each vertex C of the bicontraction H of G a PQ-tree T_C built as follows. Starting with the PQ-tree representing the clique paths of C, we replace the clique Q_e^C by e for each edge e incident to C in H (in case two such cliques coincide we introduce a P-node). We further reduce this tree to account for blocks and parallel edges in H (for brevity, we omit these technical details).

First, suppose that H is a 2-connected simple graph. Then, by [5], H has a planar LL-drawing if and only if H is an outerplanar graph and in every outerplanar embedding of H the bounded faces form a path in the dual graph. As a planar LL-drawing, we notice that every face has exactly two vertical edges. Further, every edge that belongs to two faces is vertical. This fixes the drawing of every inner face on the path of the dual. For the end-faces, we try all possible choices for the vertical edges. This results in $O(|V(H)|^2)$ choices that completely cover all possible drawings. For each such choice, we use the PQ-trees T_C to test whether or not it corresponds to a good LL-drawing of H.

Next, if H is a simple graph but not 2-connected, we use dynamic programming to process the blocks of H. For this, similarly to [5], we distinguish special blocks in H. Let C be a vertex of H, and let K be a connected component of $H - C$. Let $\{e_1, \ldots, e_t\}$ be the edges of H between C and its neighbours in K.

We say that K is a <u>fan</u> of H <u>attached to C</u> if $Q_{e_1}^C, \ldots, Q_{e_t}^C$ are distinct cliques, and the subgraph of G corresponding to the union of $C' \in K$ is an interval graph. A fan K is a <u>*tail*</u> of H if the subgraph of G corresponding to the union of C and all $C' \in K$ is an interval graph. A fan is a <u>*proper fan*</u> if it is not a tail.

To illustrate these concepts, note that in Fig. 2 for $C = ijmqr$, the connected component $K = \{klmn, nopq\}$ of $H - C$ is a fan attached to C but it is not a tail, since $G[\{i, \ldots, r\}]$ is not an interval graph while $G[\{k, \ldots, q\}]$ is.

(3.6) *Let K be a fan of H. If H has a good LL-drawing $\mathcal{E}$, then there is a good LL-drawing $\mathcal{E}'$ of H in which all vertices of K are on the same layer.*

Let H' be obtained from H by removing all fans of H. We say that a block of H is a <u>*proper block*</u> of H if it is also a block of H'. A cutpoint C of H is a <u>*proper cutpoint*</u> of H if it belongs to two proper blocks of H, or some component of $H - C$ is a proper fan, or if at least three components of $H - C$ are tails.

For blocks B, B', we write $B \prec_{\mathcal{E}} B'$ if $e \prec_{\mathcal{E}} e'$ for all $e \in E(B)$, $e' \in E(B')$.

(3.7) *The proper blocks and proper cutpoints induce a path in the block-cutpoint tree of H. If $B_1, \ldots, B_k$ are the proper blocks on this path in this order, then either $B_1 \prec_{\mathcal{E}} \ldots \prec_{\mathcal{E}} B_k$ or $B_k \prec_{\mathcal{E}} \ldots \prec_{\mathcal{E}} B_1$ for every good LL-drawing $\mathcal{E}$ of H.*

We process the proper blocks and cutpoints of H in the order $B_1, \ldots, B_k$ given by the above claim. For each proper block B_i, we consider B_i together with all tails attached to it via non-proper cutpoints. We try all possible arrangements. It can be shown that this results in a polynomial number of choices, and we test each of them using the PQ-trees T_C, and discard those that fail the test.

Next, we test consecutive blocks B_i, B_{i+1}; let C be the cutpoint they share. We consider all possible good LL-drawings $\mathcal{E}_i$ of B_i and $\mathcal{E}_{i+1}$ of B_{i+1} that are compatible (i.e., C is on the same layer in both drawings and is the rightmost, resp. leftmost in $\mathcal{E}_i$, resp. $\mathcal{E}_{i+1}$). We try all such feasible drawings together with all attached fans and tails. In this case, we do not try all possibilities directly (since there may be exponentially many of them), but instead use the PQ-tree T_C to try them indirectly using an intersection with another PQ-tree representing our choices. As discussed earlier, this can be done in linear time. In a similar fashion, we deal with proper cutpoints that are not between proper blocks.

Finally, we deal with parallel edges which is done by incorporating additional tests that do not increase the complexity by more than a constant factor.

Now, to obtain a good LL-drawing of H, we combine good LL-drawings of compatible pairs of proper blocks and cutpoints (if possible). Since there are polynomially many choices for each block, the resulting algorithm is polynomial. We remark that the number of choices for each block can be further reduced to a constant by additional tests. This produces a procedure whose complexity is linear in the size of G. (For complete details, see the full version of the paper.)

We conclude by analyzing the total complexity of our algorithm. First, finding all bisimplicial vertices of G can be done in $O(nm)$ time by examining the neighbourhood of each vertex in G. From this, the bisplitting and the bicontraction H of G can be constructed in linear time. Also, checking for true twins in G and for loops in H is a linear time procedure, and so is [4] the intervality test on H. Similarly, constructing the PQ-trees for all vertices of H takes linear time [4], and so do all other necessary operations on these PQ-trees. Finally, our dynamic programming as described above can be also implemented to run in linear time. Hence, the overall complexity of the algorithm is $O(nm)$.

4 Conclusion

We studied recognition algorithms for special cases of B_k-VPG graphs, namely chordal B_0-VPG and 2-row B_0-VPG graphs. For both cases, we described $O(nm)$ time algorithms for recognition. The interest in these types of representations comes from applications in VLSI where they can be used to model some aspects of electrical circuits. In particular, solving the colouring problem is of interest but is unfortunately NP-complete [2] on B_k-VPG graphs for every fixed $k \geq 0$. It turns out that the problem is NP-complete already on ℓ-row B_0-VPG graphs (B_0-VPG graphs that have a representation with ℓ rows) for every fixed $\ell \geq 2$. In contrast, one can decide in linear time if an ℓ-row B_0-VPG graph can be properly coloured with t colours, when t is fixed, since in this case all yes-instances have bounded *pathwidth*, while the problem is NP-complete on B_0-VPG graphs for every fixed $t \geq 3$. In a similar vein, the independent set problem is NP-complete on B_0-VPG graphs [15] but can be solved in polynomial time on ℓ-row B_k-VPG graphs for all fixed k, ℓ. Detailed proofs are in the full version of this paper.

As a continuation of this work, it may be interesting to look at other cases where representation has bounded number of rows (three or more) or other structural restriction in order to overcome hardness of optimization problems on these representations. It should be noted that the recognition of B_k-VPG graphs for every k was recently proved to be NP-complete [12]. In that respect, it seems natural to study as well special cases of these graphs, where for instance one only allows certain types of paths in the representation. Specifically the case $k = 1$ is already of interest and is currently a subject of our ongoing research.

References

1. A. Asinowski, E. Cohen, M. C. Golumbic, V. Limouzy, M. Lipshteyn, and M. Stern, *String graphs of k-bend paths on a grid*, LAGOS 2011, in press.
2. ———, *Intersection graphs of paths on a grid*, technical report (2010).
3. M. Bandy and M. Sarrafzadeh, *Stretching a knock-knee layout for multilayer wiring*, IEEE Trans. Computing **39** (1990), 148–151.
4. K. S. Booth and G. S. Lueker, *Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms.*, Journal of Computer and System Sciences **13** (1976), 335–379.
5. S. Cornelsen, T. S., and D. Wagner, *Drawing graphs on two and three lines*, Journal of Graph Algorithms and Applications **8** (2004), 161–177.
6. D. R. Fulkerson and O. A. Gross, *Incidence matrices and interval graphs*, Pacific Journal of Mathematics **15** (1965), 835–855.
7. M. C. Golumbic, *Algorithmic graph theory and perfect graphs (2nd edition)*, North Holland, 2004.
8. M. C. Golumbic and B. Ries, *On the intersection graphs of orthogonal line segments in the plane: characterizations of some subclasses of chordal graphs*, submitted manuscript (2011).
9. B. Haeupler, K. R. Jampani, and A. Lubiw, *Testing simultaneous planarity when the common graph is 2-connected*, ISAAC 2010, LNCS 6507, 2010, pp. 410–421.
10. R. Heckmann, R. Klasing, B. Monien, and W. Unger, *Optimal embedding of complete binary trees into lines and grids*, WG 1991, LNCS 570, 1992, pp. 25–35.
11. R. E. Jamison and H. M. Mulder, *Constant tolerance intersection graphs of subtrees of a tree*, Discrete Mathematics **290** (2005), 27–46.
12. J. Kratochvíl, personal communication.
13. ———, *String graphs II. Recognizing string graphs is NP-hard*, Journal of Combinatorial Theory B **52** (1991), 67–78.
14. J. Kratochvíl and J. Matoušek, *Intersection graphs of segments*, Journal of Combinatorial Theory Series B **62** (1994), 289–315.
15. J. Kratochvíl and J. Nešetřil, *Independent set and clique problems in intersection-defined classes of graphs*, Commentationes Mathematicae Universitatis Carolinae **31** (1990), 85–93.
16. F. R. McMorris and E. R. Scheinerman, *Connectivity threshold for random chordal graphs*, Graphs and Combinatorics **7** (1991), 177–181.
17. P. Molitor, *A survey on wiring*, EIK Journal of Information Processing and Cybernetics **27** (1991), 3–19.
18. F. Sinden, *Topology of thin film circuits*, Bell System Technical Journal **45** (1966), 1639–1662.
19. D. B. West, *Introduction to graph theory (2nd edition)*, Prentice Hall, 2000.