# Drawing Clustered Graphs on an Orthogonal Grid

Peter Eades[1] and Qing-Wen Feng[2*]

[1] Department of Computer Science and Software Engineering, University of Newcastle, NSW 2308, Australia.
[2] Tom Sawyer Software, 804 Hearst Avenue, Berkeley, CA 94710, USA.

(extended abstract)

**Abstract.** Clustered graphs are graphs with recursive clustering structures over the vertices. For graphical representation, the clustering structure is represented by a simple region that contains the drawing of all the vertices which belong to that cluster. In this paper, we present an algorithm which produces planar drawings of clustered graphs in a convention known as *orthogonal grid rectangular cluster drawings*. We present an algorithm which produces such drawings with $O(n^2)$ area and with at most 3 bends in each edge. This result is as good as existing results for classical planar graphs. Further, we show that our algorithm is optimal in terms of the number of bends in each edge.

## 1 Introduction

Clustered graphs are graphs with recursive clustering structures over the vertices (see Figure 1). This type of clustering structure appears in many systems. Examples include CASE tools [39], management information systems [19], and VLSI design tools [15]. For graphical representation, the clustering structure is represented by a simple region that contains the drawing of all the vertices which belong to that cluster. Algorithms for automatically drawing of clustered graphs are difficult. Heuristic methods for drawing similar structures have been developed by Sugiyama and Misue [22, 30], North [23], and by Madden et al. [18]. Algorithms for constructing straight-line drawings of clustered graphs are given in [8, 11]; note, however, that straight-line drawings of clustered graphs can require exponential area [11]. In this paper, we present an algorithm which produce planar drawings of clustered graphs in a convention called "orthogonal grid rectangular cluster drawings". We apply a technique to order the clusters of the graph recursively, and we use the visibility representation for directed graphs to produce our drawings.

The orthogonal grid drawing convention appears in a number of applications, such as VLSI circuit design [20, 21, 37, 38] and diagrammatic interfaces for relational information systems [1, 2, 24, 27, 31]. Under the orthogonal grid drawing convention, minimizing the number of bends and minimizing the area are the

* The work described in this paper was performed when the author was studying at the Department of Computer Science and Software Engineering of the University of Newcastle.

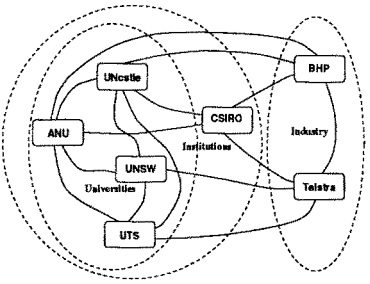



**Fig. 1.** An example of a clustered graph.

main criteria both for diagram readability and for VLSI design applications. For classical graphs, several basic results regarding planar orthogonal grid drawings have appeared in the literature. It has been shown by Valiant [38] that any planar graph of degree at most 4 admits a planar orthogonal grid drawing with area $O(n^2)$; further, there are graphs which need quadratic area. Tamassia [32] presented an $O(n^2 \log n)$ time algorithm that computes a planar orthogonal grid drawing with a given planar embedding so that the number of bends is minimized. Garg and Tamassia [14] have shown that if the planar embedding is not given, then the problem is NP-hard.

Several linear time algorithms for planar orthogonal grid drawings of classical graphs have been developed. Tamassia and Tollis [34, 35] have presented an algorithm that outputs drawings with $O(n^2)$ area, where $n$ is the number of vertices of the graph. If the graph is biconnected, then there are at most $2n + 4$ bends in the drawing; otherwise, there are at most $2.4n + 2$ bends. Further, there are at most 4 bends in each edge; if the graph is biconnected, then all but 2 edges have at most 2 bends. Kant [16, 17] has presented an algorithm which improves the result of Tamassia and Tollis in some cases. For triconnected graphs, Kant's algorithm draws on an $n \times n$ grid with at most 2 bends per edge (if $n > 6$), and the total number of bends is no more than $\lceil 3n/2 \rceil + 4$. If the graph is connected with degree at most 3, then the algorithm draws on an $\lfloor n/2 \rfloor \times \lfloor n/2 \rfloor$ grid with at most 2 bends in each edge and no more than $\lfloor n/2 \rfloor + 1$ bends in total. Even and Granot [9] have presented an algorithm such that for any planar graph with degree at most 4, the drawing has $O(n^2)$ area, and there are at most 3 bends in each edge. Lower bounds on the area and the number of bends for planar orthogonal drawings of graphs have been presented by Tamassia, Tollis and Vitter [36], and by Biedl [5]. Another useful representation for planar graphs is the *visibility representation* [28, 33]. Visibility representation is related to orthogonal drawing in that it is often used as a basis for constructing an orthogonal drawing. Several orthogonal drawing algorithms [9, 34, 35] first construct a visibility representation of the graph, then transform it to an orthogonal drawing.

In this paper, we present an algorithm for planar drawing of clustered graphs using the same approach. Our algorithm produces orthogonal grid rectangular cluster drawings with $O(n^2)$ area and with at most 3 bends in each edge. This result is as good as the results for classical planar graphs [9, 17, 35]. Further, we give an example which shows that the bend per edge performance of our

algorithm is optimal, that is, there is a class of graphs which require $\Omega(n)$ edges each

## 2 Terminology

A *clustered graph* $C = (G, T)$ consists of an undirected graph $G$ and a rooted tree $T$ such that the leaves of $T$ are exactly the vertices of $G$. Each node $\nu$ of $T$ represents a *cluster* $V(\nu)$ of the vertices of $G$ that are leaves of the subtree rooted at $\nu$. Note that tree $T$ describes an inclusion relation between clusters. In a *drawing* of a clustered graph $C = (G, T)$, graph $G$ is drawn as points and curves as usual. For each node $\nu$ of $T$, the cluster is drawn as simple closed region $R$ that contains the drawing of $G(\nu)$, such that: (1) the regions for all sub-clusters of $\nu$ are completely contained in the interior of $R$; (2) the regions for all other clusters are completely contained in the exterior of $R$; (3) if there is an edge $e$ between two vertices of $V(\nu)$ then the drawing of $e$ is completely contained in $R$. We say that the drawing of edge $e$ and region $R$ have an *edge-region crossing* if the drawing of $e$ crosses the boundary of $R$ more than once. A drawing of a clustered graph is *c-planar* if there are no edge crossings or edge-region crossings. If a clustered graph $C$ has a c-planar drawing then we say that it is *c-planar*. An edge is said to be *incident* to a cluster $V(\nu)$ if one end of the edge is a vertex of that cluster but the other endpoint is not in $V(\nu)$. An *embedding* of a clustered graph consists of the circular ordering of edges around each cluster which are incident to that cluster. A clustered graph $C = (G, T)$ is a *connected clustered graph* if each cluster induces a connected subgraph of $G$. The following results from [12] characterize c-planarity in a way which can be exploited by our drawing algorithms.

**Theorem 1.** *A connected clustered graph $C = (G, T)$ is c-planar if and only if graph $G$ is planar and there exists a planar drawing $\mathcal{D}$ of $G$, such that for each node $\nu$ of $T$, all the vertices and edges of $G - G(\nu)$ are in the external face of the drawing of $G(\nu)$.*

**Theorem 2.** *A clustered graph $C = (G, T)$ is c-planar if and only if it is a sub-clustered graph of a connected and c-planar clustered graph.*

From Theorem 2, we can assume that we are given a connected clustered graph when drawing a c-planar clustered graph. In the rest of the paper we further assume that in a clustered graph $C = (G, T)$, every nonleaf node of $T$ has at least two children.

Our techniques use the concept of planar *st*-graphs [3]. A planar *st*-graph is a planar directed graph with one source $s$ and one sink $t$; and both source and sink above can be embedded on the boundary of the same face, say the external face.

## 3 Orthogonal Drawings for C-planar Clustered Graphs

An *orthogonal grid rectangular cluster drawing (OGRC drawing)* of a clustered graph maps the graph onto a grid, where edges are drawn as sequences of horizontal and vertical segments, vertices are drawn on grid points, and region

boundaries for clusters are drawn as rectangles. In this section, we present an algorithm that produces c-planar OGRC drawings for clustered graphs. We assume that we are given a connected clustered graph $C$ with $n$ vertices, and with a c-planar embedding; further, $C$ has a sub-clustered-graph $C'$ with $n$ vertices, and every vertex of $C'$ has degree no more than 4. Our algorithm outputs c-planar OGRC drawings of $C'$ with $O(n^2)$ area, and there are at most 3 bends in each edge. We show that there is a class of c-planar embedded clustered graphs of $n$ vertices, for which every c-planar OGRC drawing requires $\Omega(n)$ edges bent more than twice. Our algorithm consists of three phases: visibility representation, orthogonalization, and bend reduction.

The result is summarized below.

**Theorem 3.** *Let $C = (G, T)$ be an $n$ vertex connected clustered graph with a c-planar embedding, and let $C'$ be an $n$ vertex sub-clustered-graph of $C$ with degree at most 4. One can construct a c-planar OGRC drawing of $C'$ with $O(n^2)$ area, and with at most 3 bends in each edge, in $O(n^2)$ time.*

**Visibility Representation** To produce a visibility representation, we first transform the clustered graph to a planar *st*-graph, taking into account the clustering structure. Then, we make use of the algorithm by di Battista, Tamassia and Tollis [4] which produces *constrained* visibility representations, that is, edges of a selected path are vertically aligned. We use this property to form rectangles for clusters. When transforming the clustered graph to a planar *st*-graph, we need to consider the clustering structure so that the visibility representation that we produce respects the clustering constraints. This is achieved by computing an *st*-numbering of the vertices of $G$ such that the vertices that belong to the same cluster are numbered consecutively. We call this numbering *c-st numbering*. The details of the construction of the c-st numbering are omitted in this extended abstract.

The *Constrained-Visibility* algorithm described in [4] takes as inputs a planar *st*-graph $G$; set $\Pi$ of paths in $G$ and produces a visibility drawing of $G$ so that the $x$-coordinates of $\Gamma(e)$ and $\Gamma(e')$ are the same whenever $e$ and $e'$ are the edges in the same path in $\Pi$. The set $\Pi$ is restricted to "nonintersecting paths" in the following sense. Two paths $\Pi_1$ and $\Pi_2$ of $G$ are said to be "nonintersecting" if they are edge disjoint and do not *cross* at common vertices, i.e., there is no vertex $v$ of $G$ with edges $e_1$, $e_2$, $e_3$ and $e_4$ incident in this clockwise order around $v$, such that $e_1$ and $e_3$ are in $\Pi_1$ and $e_2$ and $e_4$ are in $\Pi_2$.

As discussed above, with *c-st* numbering, each cluster is assigned a vertical range. To obtain the 4 bounding sides of the rectangle for a cluster $\nu$, we construct 4 dummy vertices denoted by $t(\nu)$ (top), $b(\nu)$ (bottom), $l(\nu)$ (left) and $r(\nu)$ (right); each represents one side of the rectangle. We modify $G$ recursively from bottom to top of the tree $T$, to include the dummy vertices into $G$. At each nonleaf node $\nu$ of $T$, we "bunch together" all the edges going out of $G(\nu)$ at the dummy vertex $t(\nu)$ in the external face; similarly, we bunch together all the edges coming into $G(\nu)$ at the dummy vertex $b(\nu)$ in the external face (see Figure 2). Note that there are clusters that do not have incoming or outgoing edges for the following reason. Suppose that $s$ is the single source of $G$; consider the path from $s$ to the root $\gamma$ in $T$. The construction of the *c-st* numbering ensures that for each node $\nu$ on this path, $G(\nu)$ has no incoming edges. In this case, we add an edge $(b(\nu), b(\sigma))$ to $G(\nu)$, where $\sigma$ is the child of $\nu$ on the path

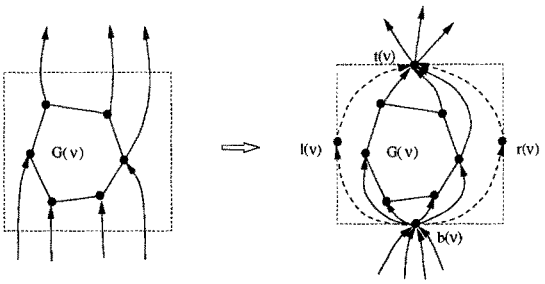



**Fig. 2.** Modify $G(\nu)$, adding dummy vertices to represent the rectangle.

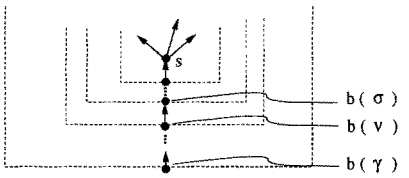



**Fig. 3.** Connect the dummy vertices to the source.

of $T$ between the root $\gamma$ and $s$ (see Figure 3). Note that the dummy vertex $b(\sigma)$ has already been included into $G(\sigma)$ by recursion. If $\sigma = s$, then we add edge $(b(\nu), s)$ instead. Similarly, if $G(\nu)$ does not have an outgoing edge, then we add edge $(t(\tau), t(\nu))$ to $G(\nu)$, where $\tau$ is the child of $\nu$ on the path of $T$ between the root $\gamma$ and $t$. If $\tau = t$, then we add edge $(t, t(\nu))$ instead. After this, $G(\nu)$ becomes a planar subgraph with a single source $b(\nu)$ and single sink $t(\nu)$. Further, we add two dummy paths $(b(\nu), l(\nu), t(\nu))$ and $(b(\nu), r(\nu), t(\nu))$ in the left face and the right face of $G(\nu)$ respectively (see Figure 2). Clearly, the modified $G(\nu)$ is a planar $st$-subgraph with source $b(\nu)$ and sink $t(\nu)$, and with external face bounded by two directed paths $(b(\nu), l(\nu), t(\nu))$ and $(b(\nu), r(\nu), t(\nu))$. We modify every subgraph $G(\nu)$ recursively as above, obtaining an extended graph $F$ that includes the dummy vertices for the rectangles. Clearly, by our construction, the resulting graph $F$ is a planar $st$-graph. Assuming that every nonleaf node of $T$ has at least two children, then the tree $T$ has less than $2n$ nodes. Since we add 4 dummy vertices for every nonleaf node of $T$, the resulting graph $F$ has $O(n)$ vertices. Note that each intercluster edge in $G$ is replaced by a path in $F$, with vertices on the path representing the region boundaries it crosses (see Figure 4). Therefore, if $G$ has $n$ vertices and hence $O(n)$ edges, then graph $F$ has $O(n^2)$ edges ($F$ may have multiple edges).

We specify the following alignment requirements in $F$ for our visibility representation. For each nonleaf node $\nu$ of $T$, we require that the edges on each of the dummy paths $(b(\nu), l(\nu), t(\nu))$ and $(b(\nu), r(\nu), t(\nu))$ are aligned. For each path of $F$ that represents an intercluster edge, we require that all the edges on the path are aligned. To avoid introducing unnecessary bends in the orthogo-

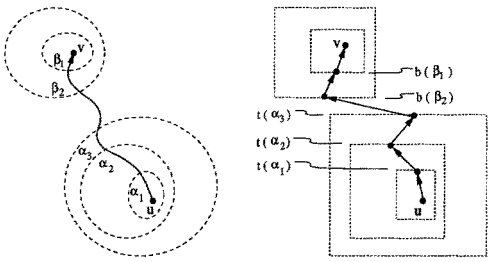



**Fig. 4.** An edge in $G$ is replaced by a path.

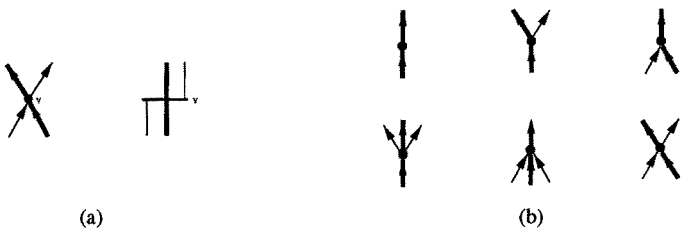



**Fig. 5.** (a) Alignment requirement for a vertex $v$. (b) Edge alignment rules.

nalization phase, we require that some edges around a vertex are aligned. For example, suppose that there is a vertex $v$ that has two incoming edges and two outgoing edges, then we require that the right incoming edge is aligned with the left outgoing edge (see Figure 5(a)). Note that we are drawing a sub-clustered-graph $C'$ which has degree at most 4, some of the edges in $G$ will be removed in the final drawing. Here we only need to consider those edges that are in the sub-clustered-graph, and note that each vertex has at most 4 edges incident to it in the sub-clustered-graph. Figure 5(b) illustrates all the cases for our alignment requirements; the edges that are marked by thick lines are required to be aligned (edges that do not belong to the sub-clustered-graph are not shown here). All the above alignment requirements together form a complete specification of the paths that are to be aligned for our visibility representation. Although some of these paths share common vertices, they do not intersect with each other. This is because there is at most one path going through each original (nondummy) vertex of $G$, and at every dummy vertex, the paths originate from distinct edges of $G$ and therefore do not intersect. We apply the algorithm *Constrained_Visibility_Draw* to $F$ with the above requirements for alignment. Thus we obtain a visibility representation $\Gamma$ of $F$ with the vertices and edges of each cluster $\nu$ drawn within a rectangle formed by $\Gamma(t(\nu))$, $\Gamma(b(\nu), l(\nu), t(\nu))$, $\Gamma(b(\nu))$ and $\Gamma(b(\nu), r(\nu), t(\nu))$ as in Figure 6. The extended graph $F$ has $O(n)$ vertices and $O(n^2)$ edges and thus this step takes time $O(n^2)$; it produces a visibility drawing on a grid of size $O(n) \times O(n)$.

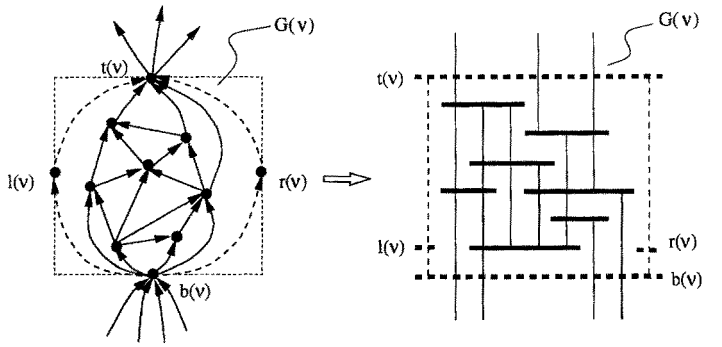



**Fig. 6.** Forming a rectangle for a cluster.

**Orthogonalization** In this phase, we first remove those edges which do not belong to the sub-clustered-graph, but we retain the dummy vertices and the dummy paths for the rectangles. To transform the visibility representation to an OGRC drawing, we only need to perform some local operations at each vertex, transforming a horizontal segment to a point. These local operations are illustrated in Figure 7; symmetric cases for (a), (c), (d), (e) and (f) are omitted for simplicity. Note that Figure 7 covers all the cases that can appear, since we have required that certain edges around a vertex are aligned. Further, note that a new row is added at every source or sink of degree 4 (see Figure 7(h) and (i)).

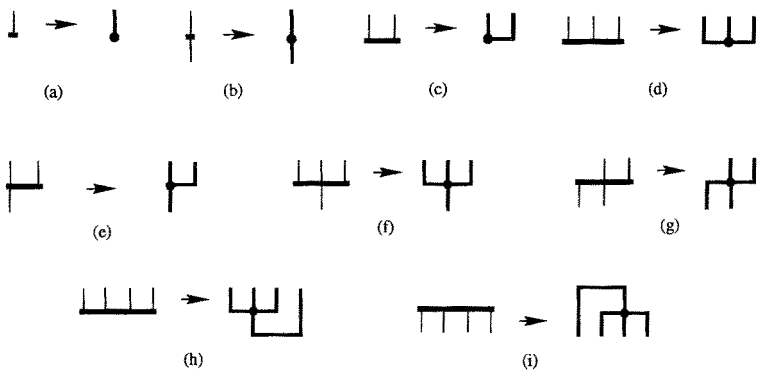
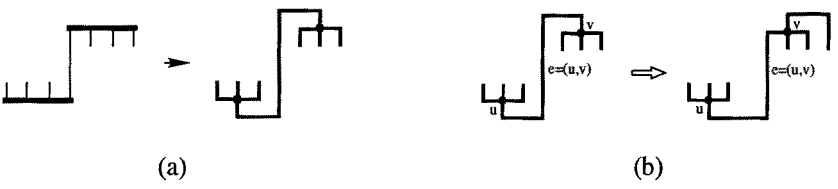



**Fig. 7.** Orthogonalization rules.

**Bend Reduction** In the OGRC drawing obtained from the previous phase, an edge is bent only near its endpoints. Hence every edge can have at most 3




(a) (b)

**Fig. 8.** (a) An edge with 4 bends. (b) Reduce a bend on edge $e$.

bends except in the following case: the edge is between a source of degree 4 and a sink of degree 4 and it has 2 bends near each endpoint (see Figure 8(a)). We show that this 4 bend case can be eliminated by making some adjustments in the drawing. Note that for a source or sink $u$ of degree 4, either the leftmost edge or the rightmost edge gets 2 bends near it, but not both. We call the leftmost or the rightmost edge of $u$ an *extreme edge* of $u$. Suppose that there is an edge $e = (u, v)$ that has 2 bends near $u$ and another 2 bends $v$. Then we fix one end $u$ and rotate the edges around the other end $v$, letting the edge $e$ bend only once near $v$ (see Figure 8(b)). After this, we say that the edge $e$ is "processed", and the vertices $u$ and $v$ are "visited". However, this operation creates a new bend in the other extreme edge $e' = (v, w)$ of $v$. This may cause some problems if vertex $w$ is a source or sink of degree 4 and $e'$ is one of its extreme edges. In this case, we continue to process edge $e'$ with the visited endpoint $v$ fixed. We repeat this until we meet an edge $e' = (v, w)$ such that either $e'$ is not an extreme edge of $w$, or $w$ has already been visited. Clearly, if $e'$ is not an extreme edge of $w$, then it has at most 3 bends. If $w$ has already been visited, then there is a processed edge incident to $w$ that gets two bends near $w$, therefore $e'$ has at most 1 bend near $w$ and hence can have at most 3 bends. We keep performing this until there are no 4 bend edges exist.

The bend reduction step takes $O(n^2)$ time. Details are omitted from this extended abstract.

## 4 A Lower Bound for Bends

In this section, we present a class of c-planar embedded clustered graphs of $n$ vertices, for which every c-planar OGRC drawing requires $\Omega(n)$ edges each bent more than twice. This shows that our algorithm *Orthogonal_Grid_Rectangular_Draw* is optimal in the worst case (in terms of the number of bends in each edge). To prove our result, let us first consider a small sub-clustered-graph $I$ (see Figure 9), which serves as the building block of our cluster graph. There are two clusters $A$ and $B$ (see Figure 9) in the sub-clustered-graph $I$. Cluster $A$ contains vertices $a_1, a_2, \ldots, a_7$; cluster $B$ contains vertices $b_1, b_2, \ldots, b_4$. We assume that $I$ has a fixed c-planar embedding; the orderings of the edges around cluster $A$ and cluster $B$ are shown in Figure 10. The drawings that we discuss in the rest of this section are all consistent with this embedding. We prove the following lemma.

**Lemma 4.** *In every c-planar OGRC drawing of $I$, there is at least one edge bent more than twice.*

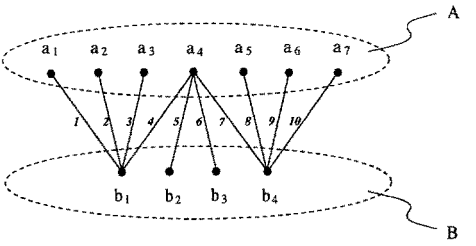



**Fig. 9.** The sub-clustered-graph $I$.

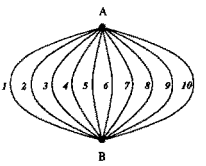



**Fig. 10.** The embedding for clusters $A$ and $B$.

*Proof.* In a c-planar OGRC drawing of $I$, cluster $A$ and $B$ are drawn as disjoint rectangular regions. Without loss of generality, we assume that the rectangle for $A$ is drawn above the rectangle for $B$, and there is a horizontal line $\ell$ separating them. Note that all the edges between $A$ and $B$ have to cross this horizontal line $\ell$, and they cross the line $\ell$ in the order shown in Figure 10. Consider the edge $(a_4, b_1)$ and the edge $(a_4, b_4)$ both incident to $a_4$. We show that at least one of these two edges has more than one bend above the line $\ell$. If the edge $(a_4, b_1)$ has no bend above the line $\ell$, then the other edge $(a_4, b_4)$ must have at least 3 bends above the line $\ell$; if the edge $(a_4, b_1)$ has one bend above the line $\ell$, then the other edge $(a_4, b_4)$ must have at least 2 bends above the line $\ell$. With out loss of generality, let us assume that the edge $(a_4, b_4)$ has more than one bend above the line $\ell$. Now consider the edge $(a_4, b_4)$ together with the edge $(a_7, b_4)$. We show that at least one of them has a total of more than two bends. If the edge $(a_4, b_4)$ has at least one bend below the line $\ell$, then it has more than two bends in total; if the edge $(a_4, b_4)$ has no bend below the line $\ell$, then the other edge $(a_7, b_4)$ must have more than two bends below the line $\ell$. Therefore, we have that there is at least one edge in $I$ that has a total of more than two bends. □

Now we define a class of clustered graphs $\Phi_n$ $(n = 1, 2, \ldots)$ with sub-clustered-graph $I$ as the building block. Clustered graph $\Phi_n$ consists of a sequence of $n$ copies of the sub-clustered-graph $I$ (see Figure 11). The vertex $a_7$ of a previous copy of $I$ also serves as the vertex $a_1$ of the next copy of $I$. Clustered graph $\Phi_n$ has two clusters $A_n$ and $B_n$. Cluster $A_n$ contains the vertices in the cluster $A$ of each sub-clustered-graph $I$; cluster $B_n$ contains the vertices in the cluster $B$ of each sub-clustered-graph $I$. Clearly, $\Phi_n$ has $10n + 1$ vertices. By Lemma 4, we have the following theorem.

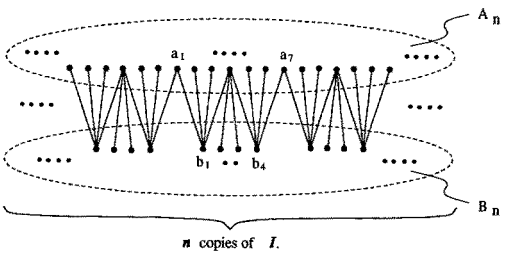



**Fig. 11.** The construction of the clustered graph $\Phi_n$.

**Theorem 5.** *In every c-planar OGRC drawing of $\Phi_n$ $(n = 1, 2, \ldots)$, there are at least n edges bent more than twice.*

## 5 Remarks

In this paper, we present an algorithm *Orthogonal_Grid_Rectangular_Draw* that produces c-planar OGRC drawings with $O(n^2)$ area and with at most 3 bends in each edge. This result is as good as the results for classical planar graphs [9, 17, 35]. Lower bounds for the area of orthogonal drawings of classical graphs [38] imply that the area of the drawing produced by our algorithm is asymptotically optimal. Further, we show that the bend per edge performance of our algorithm is optimal. Nevertheless, some open problems remain:

- Although the height and the width of our output drawings are both $O(n)$, our algorithm does not guarantee a good aspect ratio. In practice, our algorithm may produce drawings which clearly prefer one dimension against the other; this is because we use a visibility representation which is biased to one dimension. Recently, some study has been done on the problem of "2-dimensional visibility representations" [7, 13] of planar graphs. In a 2-dimensional visibility representation, each vertex is represented by a box and each edge is represented by a horizontal or vertical segment between the sides of the boxes. It would be interesting to study how to use 2-dimensional visibility representations to improve our algorithm, especially in terms of the aspect ratio of the drawing.
- Even and Granot [10] have presented some algorithms for grid layout of block diagrams. Although the drawing requirements there are different from the requirements of drawing clustered graphs, it would be worthwhile to investigate whether we can borrow some of the techniques there and use them in drawing clustered graphs.
- In recent years, many results have been achieved for orthogonal drawings of non planar graphs [6, 5, 25, 26]. It seems very profitable to use these results to extend our algorithm to non planar clustered graphs.

## References


1. C. Batini, L. Furlani, and E. Nardelli. What is a good diagram? a pragmatic approach. In *Proc. 4th Int. Conf. on the Entity Relationship Approach*, 1985.
2. C. Batini, M. Talamo, and R. Tamassia. Computer aided layout of entity-relationship diagrams. *Journal of Systems and Software*, 4:163–173, 1984.
3. G. di Battista and R. Tamassia. Algorithms for plane representations of acyclic digraphs. *Theoretical Computer Science*, 61:175–198, 1988.
4. G. di Battista, R. Tamassia, and I.G. Tollis. Constrained visibility representations of graphs. *Information Processing Letters*, 41:1–7, 1992.
5. T. Biedl. New lower bounds for orthogonal graph drawings. In Franz J. Brandenburg, editor, *GD'95*, volume 1027 of *Lecture Notes in Computer Science*, pages 28–39. Springer-Verlag, 1995.
6. T. Biedl and G. Kant. A better heuristic for orthogonal graph drawings. In *ESA'94*, volume 855 of *Lecture Notes in Computer Science*, pages 24–35. Springer-Verlag, 1994.
7. P. Bose, A. Dean, J. Hutchinson., and T. Shermer. On rectangle visibility graphs. In Stephen C. North, editor, *GD'96*, volume 1190 of *Lecture Notes in Computer Science*, pages 25–44. Springer-Verlag, 1997.
8. P. Eades, Q. Feng, and X. Lin. Straight-line drawing algorithms for hierarchical graphs and clustered graphs. In Stephen C. North, editor, *GD'96*, volume 1190 of *Lecture Notes in Computer Science*, pages 113–128. Springer-Verlag, 1997.
9. S. Even and G. Granot. Rectilinear planar drawings with few bends in each edge. Technical Report 797, Computer Science Department, Technion, Israel Institute of Technology, 1994.
10. S. Even and G. Granot. Grid layout of block diagrams - bounding the number of bends in each connection. In R. Tamassia and I. G. Tollis, editors, *GD'94*, volume 894 of *Lecture Notes in Computer Science*, pages 64–75. Springer-Verlag, 1995.
11. Q. Feng, R. Cohen, and P. Eades. How to draw a planar clustered graph. In *COCOON'95*, volume 959 of *Lecture Notes in Computer Science*, pages 21–31. Springer-Verlag, 1995.
12. Q. Feng, R. Cohen, and P. Eades. Planarity for clustered graphs. In *ESA'95*, volume 979 of *Lecture Notes in Computer Science*, pages 213–226. Springer-Verlag, 1995.
13. U. Fößmeier, G. Kant, and M. Kaufmann. 2-visibility drawings of planar graphs. In Stephen C. North, editor, *GD'96*, volume 1190 of *Lecture Notes in Computer Science*, pages 155–168. Springer-Verlag, 1997.
14. A. Garg and R. amassia. On the computational complexity of upward and rectilinear planarity testing. In R. Tamassia and I. G. Tollis, editors, *GD'94*, volume 894 of *Lecture Notes in Computer Science*, pages 286–297. Springer-Verlag, 1995.
15. D. Harel. On visual formalisms. *Communications of the ACM*, 31(5):514–530, 1988.
16. G. Kant. Drawing planar graphs using the *lmc*-ordering. In *Proc. 33th IEEE Symp. on Foundations of Computer Science*, pages 101–110, 1992.
17. G. Kant. Drawing planar graphs using the canonical ordering. *Algorithmica*, 16:4–32, 1996.
18. G. Kar, B.P. Madden, and R.S. Gilbert. Heuristic layout algorithms for network management presentation services. *IEEE Network*, pages 29–36, November 1988.
19. J. Kawakita. The KJ method – a scientific approach to problem solving. Technical report, Kawakita Research Institute, Tokyo, 1975.

20. M.R. Kramer and J. van Leeuwen. The complexity of wire-routing and finding minimum area layouts for arbitrary VLSI circuits. In F.P. Preparata, editor, *Advances in Computing Research*, volume 2, pages 129–146. JAI Press, Greenwich, Conn., 1985.
21. C. E. Leiserson. Area-efficient graph layouts (for VLSI). In *Proceedings of the IEEE Symposium on the Foundations of Computer Science*, pages 270 – 281, 1980.
22. K. Misue and K. Sugiyama. An overview of diagram based idea organizer: D-abductor. Technical Report IIAS-RR-93-3E, ISIS, Fujitsu Laboratories, 1993.
23. S. North. Drawing ranked digraphs with recursive clusters. In *Proc. ALCOM Workshop on Graph Drawing '93*, September 1993.
24. J. Nummenmaa and J. Tuomi. Constructing layouts for er-diagrams from visibility representations. In *Proc. 9th Int. Conf. on Entity-Relationship Approach*, pages 303–317, 1990.
25. A. Papakostas and I. G. Tollis. Improved algorithms and bounds for orthogonal drawings. In R. Tamassia and I. G. Tollis, editors, *GD'94*, volume 894 of *Lecture Notes in Computer Science*, pages 40–51. Springer-Verlag, 1994.
26. A. Papakostas and I. G. Tollis. A pairing technique for area-efficient orthogonal drawings. In Stephen C. North, editor, *GD'96*, volume 1190 of *Lecture Notes in Computer Science*, pages 355–370. Springer-Verlag, 1997.
27. D. Reiner, G. Brown, M. Friedell, J. Lehman, R. McKee, P. Rheingans, and A. Rosenthal. A database designer's workbench. In S. Spaccapietra, editor, *Entity-Relationship Approach: Proc. 5th Int. Conf. on Entity-Relationship Approach (Dijon France 1987)*, pages 347–360, New York, N.Y., 1987. North-Holland.
28. P. Rosenstiehl and R.E. Tarjan. Rectilinear planar layouts and bipolar orientations of planar graphs. *Discrete and Computational Geometry*, 1(4):343–353, 1986.
29. J.A. Storer. On minimal node-cost planar embeddings. *Networks*, 14:181–212, 1984.
30. K. Sugiyama and K. Misue. Visualization of structural information: Automatic drawing of compound digraphs. *IEEE Transactions on Software Engineering*, 21(4):876–892, 1991.
31. R. Tamassia. New layout techniques for entity-relationship diagrams. In *Proc. 4th Int. Conf. on Entity-Relationship Approach*, pages 304–311, 1985.
32. R. Tamassia. On embedding a graph in the grid with the minimum number of bends. *SIAM J. Computing*, 16(3):421–444, 1987.
33. R. Tamassia and I.G. Tollis. A unified approach to visibility representations of planar graphs. *Discrete and Computational Geometry*, 1(4):321–341, 1986.
34. R. Tamassia and I.G. Tollis. Efficient embedding of planar graphs in linear time. In *Proc. IEEE Int. Symp. on Circuits and Systems*, pages 495–498, 1987.
35. R. Tamassia and I.G. Tollis. Planar grid embedding in linear time. *IEEE Trans. on Circuits and Systems*, CAS-36(9):1230–1234, 1989.
36. R. Tamassia, I.G. Tollis, and J.S. Vitter. Lower bounds for planar orthogonal drawings of graphs. *Information Processing Letters*, 39:35–40, 1991.
37. J.D. Ullman. *Computational Aspects of VLSI.* Principles of Computer Science. Computer Science Press, Rockville, Md., 1984.
38. L. Valiant. Universality considerations in VLSI circuits. *IEEE Transactions on Computers*, C-30(2):135–140, 1981.
39. C. Williams, J. Rasure, and C. Hansen. The state of the art of visual languages for visualization. In *Visualization 92*, pages 202 – 209, 1992.