# A More Compact Visibility Representation

Goos Kant
Dept. of Computer Science
Utrecht University
Padualaan 14
3584 CH Utrecht
the Netherlands
goos@cs.ruu.nl

RUU-CS-93-26
August 1993

Utrecht University
Department of Computer Science
Padualaan 14, P.O. Box 80.089,
3508 TB Utrecht, The Netherlands,
Tel. : + 31 - 30 - 531454

# A More Compact Visibility Representation

Goos Kant
Dept. of Computer Science
Utrecht University
Padualaan 14
3584 CH Utrecht
the Netherlands
goos@cs.ruu.nl

Technical Report RUU-CS-93-26
August 1993

Department of Computer Science
Utrecht University
P.O.Box 80.089
3508 TB Utrecht
The Netherlands

ISSN: 0924–3275

# A More Compact Visibility Representation*

Goos Kant
Dept. of Computer Science
Utrecht University
Padualaan 14
3584 CH Utrecht
the Netherlands
goos@cs.ruu.nl

**Abstract**

In this paper we present a linear time and space algorithm for constructing a visibility representation of a planar graph on an $(\lfloor \frac{3}{2}n \rfloor - 3) \times (n-1)$ grid, thereby improving the previous bound of $(2n-5) \times (n-1)$. To this end we build in linear time the 4-block tree of a triangulated planar graph.

## 1 Introduction

The problem of "nicely" drawing a graph in the plane has received increasing attention due to the large number of applications [3]. Examples include VLSI layout, algorithm animation, visual languages and CASE tools. Several criteria to obtain a high aesthetic quality have been established. Typically, vertices are represented by distinct points in a line or plane, and are sometimes restricted to be grid points. (Alternatively, vertices are sometimes represented by line segments.) Edges are often constrained to be drawn as straight lines or as a contiguous set of line segments (e.g., when bends are allowed). The objective is to find a layout for a graph that optimizes some cost function, such as area, minimum angle, number of bends, or satisfies some other constraint (see [3] for an up to date overview).

One of the most beautiful ways for drawing $G$ is by using a *visibility representation*. In a visibility representation every vertex is mapped to a horizontal segment, and every edge is mapped to a vertical line, only touching the two vertex segments of its endpoints. It is clear that this leads to a nice and readable picture, and it

*This work was supported by ESPRIT Basic Research Action No. 7141 (project ALCOM II: *Algorithms and Complexity*). Part of this work was done while visiting the Graph Theory workshop at the Bellairs Research Institute of McGill University (Montreal), Feb. 12-19, 1993.

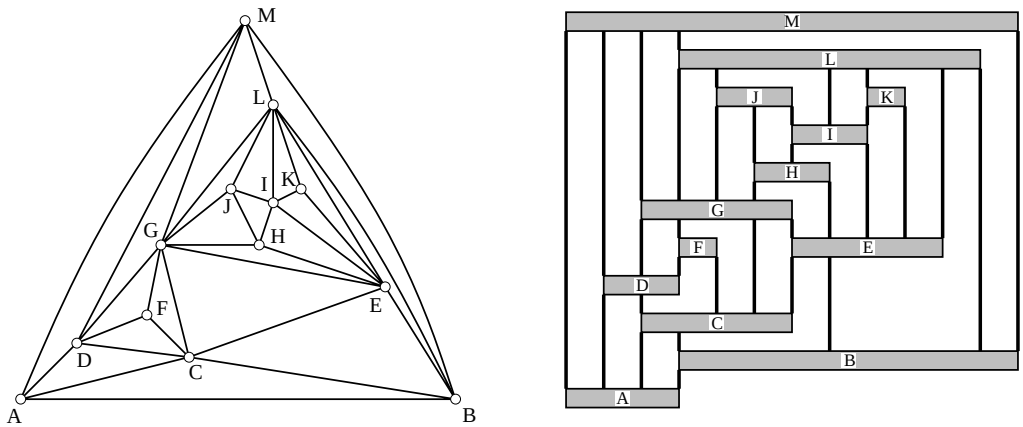



Figure 1: Example of a visibility representation.

therefore gains a lot of interest. It has been applied in several industrial applications, for representing electrical diagrams and schemas (Rosenstiehl, personal communication). See Figure 1 for an example. Otten & Van Wijk [15] showed that every planar graph admits such a representation, and a linear time algorithm for constructing it is given by Rosenstiehl & Tarjan [18] (independently, Tamassia & Tollis [19] came up with the same algorithm). The size of the required grid is $(2n-5) \times (n-1)$, with $n$ the number of vertices. The algorithm is based on a so called *st-numbering*: a numbering $v_1, \ldots, v_n$ of the vertices such that $(v_1, v_n) \in G$ and every vertex $v_i$ $(1 < i < n)$ has neighbors $v_j$ and $v_k$ with $j < i < k$. The *height* of the drawing is the longest path from $v_1$ to $v_n$, which has length at most $n-1$. The *width* of the drawing is the longest path in the dual graph, which is $f-1$, where $f$ is the number of faces in $G$ (by Euler's formula: $m \leq 3n-6$ and $f = m-n+2$).

The algorithm is used in several drawing algorithms. We mention here the algorithm of Tamassia & Tollis [20] for constructing an orthogonal drawing, and the work of Di Battista, Tamassia & Tollis [5] for computing *constrained* visibility representations. Rosenstiehl & Tarjan also discuss the open problems concerning the grid size of visibility representations [18]. The requirement of using a small area seems to become a core area in the research field of graph drawing, due to the important applications in VLSI-design and chip layout (e.g., see the titles and contents of [1, 4, 10]).

In this paper we show that every planar graph can be represented by a visibility representation on a grid of size at most $(\lfloor \frac{3}{2}n \rfloor - 3) \times (n-1)$. This improves all previous bounds considerably. An outline of the algorithm to achieve this is as follows. Assume the input graph $G$ is triangulated (otherwise a simple linear time algorithm can be applied to make it so [12, 16]). Then we split $G$ into its 4-connected components, and construct the 4-block tree of $G$. We show that we can do this in

linear time for triangulated planar graphs, thereby improving the $O(n \cdot \alpha(m,n)+m)$ time algorithm of Kanevsky et al. [9] for this special case. To each 4-connected component the algorithm of Kant & He is applied, who showed that if the planar graph is 4-connected, then a visibility representation of it can be constructed with grid size at most $(n-1) \times (n-1)$ [13]. The representations of the 4-connected components are combined into one entire drawing, leading to the desired width.

The paper is organized as follows. In Section 2 we deliver the necessary definitions, theorems and the algorithm for constructing a visibility representation. In Section 3 we give our framework for computing a more compact visibility representation. In Section 4 we show that constructing a 4-block tree of a triangulated planar graph can be constructed in linear time. Section 5 contains some final remarks and open questions.

# 2 Definition and Backgrounds

Let $G = (V, E)$ be a planar graph with $n$ vertices and $m$ edges. A graph is called *planar* if it can be drawn without any pair of crossing edges. A *planar embedding* is a representation of a planar graph in which at every vertex all edges are sorted in clockwise order when visiting them around the vertex with respect to the planar drawing. The embedding divides the plane into a number of *faces*. The unbounded face is the *exterior face* or *outerface*. A cycle of length 3 is a *triangle*. A planar graph is *triangulated* if every face is a triangle. A triangulated planar graph has $3n-6$ edges and adding any edge to it destroys the planarity. A cycle $C$ of $G$ divides the plane into its interior and exterior region. If $C$ contains at least one vertex in its interior and its exterior, then $C$ is called a *separating cycle*. A graph $G$ is called *k-connected*, if deleting of any $k-1$ vertices does not disconnect $G$. In our algorithms we need the following theorem of Kant & He.

**Theorem 2.1 ([13])** *There exists a labeling of the vertices $v_1 = u, v_2 = v, v_3, \ldots, v_n = w$ of a triangulated 4-connected planar graph $G$ with outerface $u, v, w$, meeting the following requirements for every $4 \leq k \leq n$:*

1. *The subgraph $G_{k-1}$ of $G$ induced by $v_1, v_2, \ldots, v_{k-1}$ is 2-connected and the boundary of its exterior face is a cycle $C_{k-1}$ containing the edge $(u, v)$.*

2. *$v_k$ is in the exterior face of $G_{k-1}$, and its neighbors in $G_{k-1}$ form a (at least 2-element) subinterval of the path $C_{k-1} - \{(u, v)\}$. If $k \leq n-2$, $v_k$ has at least 2 neighbors in $G - G_{k-1}$.*

We will call this ordering the *canonical 4-ordering*. It is shown in [13] that a canonical 4-ordering can be constructed in linear time for a 4-connected triangular planar graph $G$. Moreover, the ordering can be made such that $v_{n-1}$ is a neighbor of both $v_1$ and $v_n$. Every canonical 4-ordering of $G$ is also an $st$-numbering of $G$.

Let all edges $(v_i, v_j)$ be directed $v_i \to v_j$ if $j > i$. Then all incoming edges of any vertex $v$ in $G$ appear consecutively around $v$, as do all outgoing edges, in any planar embedding of $G$. $in(v)$ and $out(v)$ denote the number of incoming and outgoing edges of $v$, respectively. Let $v$ be a vertex with incoming edges from vertices $u_1, \ldots, u_{in(v)}$ (from left to right) and outgoing edges to vertices $w_1, \ldots, w_{out(v)}$. We call $u_1$ the *leftvertex* of $v$ and $u_{in(v)}$ the *rightvertex* of $v$. $w_1$ is called the *leftup* of $v$ and $w_{out(v)}$ is called the *rightup* of $v$ (see Figure 2). The following lemma is useful:

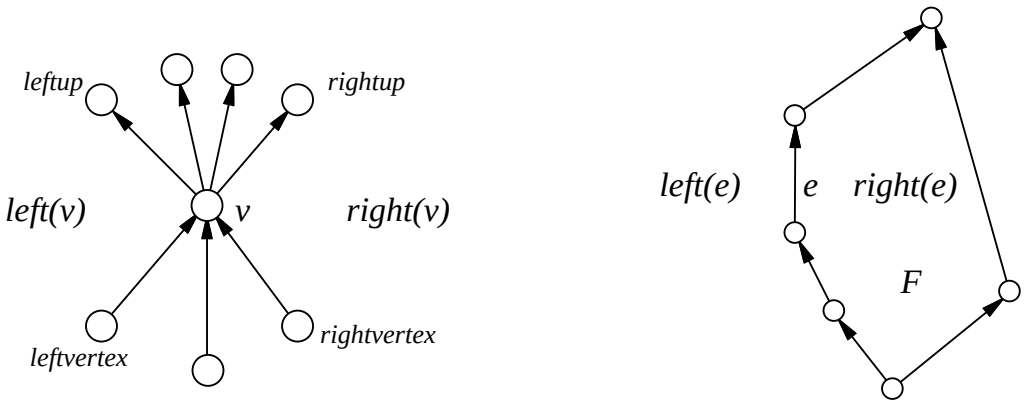



Figure 2: Properties of planar $st$-graphs.

**Lemma 2.2** *Let $v_1, v_2, \ldots, v_n$ be a canonical 4-ordering of a 4-connected triangular planar graph $G$, such that $v_{n-1}$ is a neighbor of both $v_1$ and $v_n$. Then the numbering $u_i = v_{n-i+1}$ (with $1 \leq i \leq n$) is also a correct canonical 4-ordering $u_1, \ldots, u_n$ of $G$.*

**Proof:** By 4-connectivity of $G$, $v_1, v_{n-1}$ and $v_n$ form one face in $G$, hence the vertices $u_1, u_2$ and $u_n$ are forming one face. Every vertex $u_i$ $(2 < i < n-1)$ has at least 2 incoming and 2 outgoing edges, since vertex $v_{n-i+1}$ has at least 2 outgoing and 2 incoming edges. From this observation also the 2-connectivity of $G_i$, the induced subgraph on $u_1, \ldots, u_i$, follows, which completes the proof. □

The boundary of every face consists of exactly two directed paths in $G$ [18, 19]. We define *left*$(e)$ (*right*$(e)$) to be the face to the left (right) of $e$. The face separating the incoming from the outgoing edges in the clockwise direction is called *left*$(v)$ and the other separating face is called *right*$(v)$.

Vertex $v_1$ has no incoming edges (source) and $v_n$ has no outgoing edges (sink). Let $d(v)$ denote the length of the longest path from $v_1$ to $v$. Let $D = d(v_n)$. $G^*$ denotes the dual graph of $G$: every face of $G$ is directed by a vertex in $G^*$, and there is an edge between two vertices in $G^*$, if the corresponding faces in $G$ share an edge. We direct the edges of $G^*$ as follows: if $F_l$ and $F_r$ are the left and the right face of

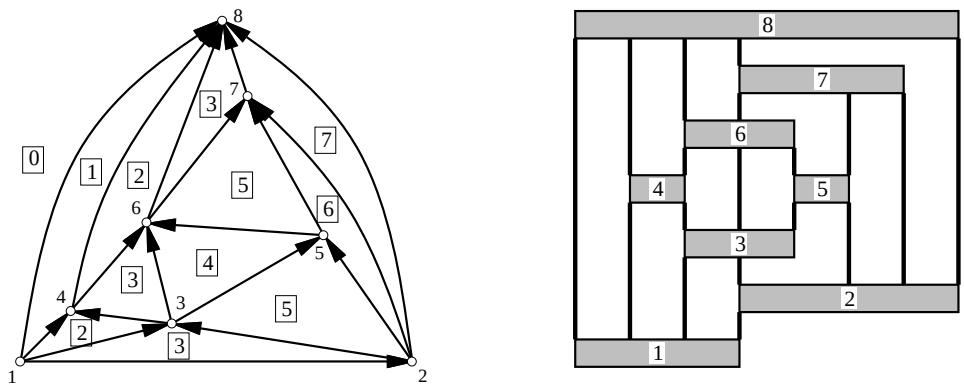



Figure 3: The canonical 4-ordering and corresponding visibility representation.

some edge $(v, w)$ of $G$, direct the dual edge from $F_l$ to $F_r$ if $(v, w) \neq (v_1, v_n)$ and from $F_r$ to $F_l$ if $(v, w) = (v_1, v_n)$. $G^*$ is also a planar $st$-graph. For each node $F$ of $G^*$, let $d^*(F)$ denote the number of *nodes* on the longest path from the source node of $G^*$ to $F$. Let $D^*$ denote the maximum of $d^*(F)$. The algorithm for constructing the visibility representation of Rosenstiehl & Tarjan [18] and Tamassia & Tollis [19] can now be described as follows:

VISIBILITY($G$);
  compute an $st$-numbering for $G$;
  construct the planar $st$-graph $G$ and its dual $G^*$;
  compute $d(v)$ for all vertices in $G$ and $d^*(F)$ for all vertices in $G^*$;
  **for** each vertex $v \neq s, t$ **do**
    draw a horizontal line between $(d^*(\mathit{left}(v)), d(v))$ and $(d^*(\mathit{right}(v)) - 1, d(v))$;
  **rof**;
  for vertex $s$, draw a horizontal line between $(0, 0)$ and $(D^* - 1, 0)$;
  for vertex $t$, draw a horizontal line between $(0, D)$ and $(D^* - 1, D)$;
  **for** each edge $(u, v) \neq (s, t)$ **do**
    draw a line between $(d^*(\mathit{left}(u, v)), d(u))$ and $(d^*(\mathit{left}(u, v)), d(v))$;
  **rof**;
  for edge $(s, t)$, draw a line between $(0, 0)$ and $(0, D)$;
END VISIBILITY($G$);

In the remaining part of this paper, , $(G)$ denotes the drawing, obtained by applying VISIBILITY($G$). $y(v)$ denotes the $y$-coordinate of the segment of vertex $v$, and $x(u, v)$ denotes the $x$-coordinate of edge $(u, v)$ in , $(G)$. Notice that $x(v_1, v_n) < x(v_1, v_2) < x(v_2, v_n)$ in , $(G)$. The size of the drawing is the size of the smallest rectangle with sides parallel to the $x$- and $y$-axis that covers the drawing. See Figure 3 for a 4-connected triangular planar graph $G$ with a canonical 4-ordering,

and the corresponding visibility representation , $(G)$. (The small squares in the figure represent the vertices of $G^*$ and the integers in the squares represent their $d^*$ values.)

**Theorem 2.3 ([13])** *If $G$ is a 4-connected triangular planar graph and we take the canonical 4-ordering as st-ordering, then the algorithm* VISIBILITY *constructs a visibility representation of $G$ on a grid of size at most $(n-1) \times (n-1)$ in linear time and space.*

# 3 A General Compact Visibility Representation

In this section we show how we can construct a visibility representation of a planar graph $G$ on a grid, yielding a width smaller than $2n-5$. We assume that $G$ is triangulated (otherwise apply an arbitrary linear time and space triangulation algorithm [12]). To apply the result of Theorem 2.3 we split $G$ into its 4-connected components. Since $G$ is triangulated, a 4-connected component of $G$ is a 4-connected triangulated planar subgraph of $G$. From this we construct the *4-block tree* $T$ of $G$: every 4-connected component $G_b$ of $G$ is represented by a node $b$ in $T$. There is an edge between two nodes $b$ and $b'$ in $T$, if the separating triplet belongs to both $G_b$ and $G_{b'}$. By planarity every triplet of three vertices is a separating triangle and belongs to precisely two 4-connected components. The separating triplet is an interior face in $G_b$ and the exterior face of $G_{b'}$. We show in Section 4 that $T$ can be computed in linear time and space for triangulated planar graphs. See Figure 4 for the 4-block tree of the graph in Figure 1.

We root $T$ at an arbitrary node $b$. We compute a canonical ordering for $G_b$, as defined in Theorem 2.1, and direct the edges accordingly. In the algorithm, we traverse $T$ top-down and visit the corresponding 4-connected components. Let $b'$ be a child of $b$ in $T$, and let $V(G_{b'})$ denote the set of vertices of $G_{b'}$. Let $u, v, w$ be the separating triplet (triangle) of $G_{b'}$. Assume the edges are directed $u \to v$ and $v \to w$ in $G_b$. We define $c(G_{b'}) = v$. Using Lemma 2.2, we have two possibilities for computing a canonical ordering in $G_{b'}$ (let $n' = |V(G_{b'})|$):

NORMAL$(G_{b'})$ We set $u = v_1, v = v_2$ and $w = v_{n'}$ and compute a canonical ordering $v_1, v_2, \ldots, v_n$ for $G_{b'}$,

REVERSE$(G_{b'})$ We set $u = u_{n'}, v = u_2$ and $w = u_1$, and compute the canonical ordering $u_1, u_2, \ldots, u_{n'}$ to $G_{b'}$. Then the ordering is reversed: we set $v_i := u_{n'-i+1}$, for all $i$ with $1 \leq i \leq n'$.

In NORMAL$(G_{b'})$, $v$ has number $v_2$, in REVERSE$(G_{b'})$, $v$ has number $v_{n'-1}$. In both orderings, $u = v_1$ and $w = v_{n'}$. See also Figure 5.

Both numberings will be applied in the algorithm to achieve a more compact visibility representation. We introduce a label $l(v)$ for each vertex $v$, which can have

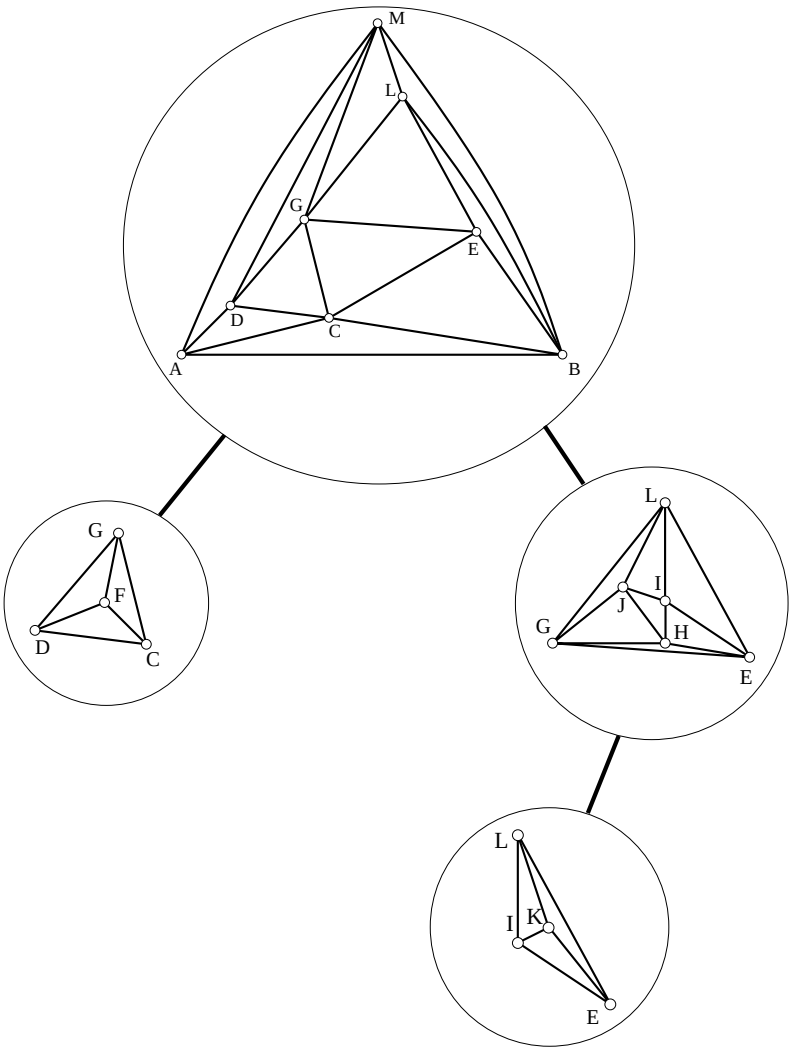



Figure 4: The 4-block tree of the graph in Figure 1.

the value *up*, *down* or *unmarked*. If $l(v) = \mathit{unmarked}$, then $v$ is called unmarked, otherwise $v$ is called marked. Assume we visit node $b'$ in $T$, and we have to compute a canonical ordering for $G_{b'}$. Let $v = c(G_{b'})$. The value of $l(v)$ implies whether we use $\textsc{Normal}(G_{b'})$ or $\textsc{Reverse}(G_{b'})$: if $l(v) = \mathit{up}$, we apply $\textsc{Normal}(G_{b'})$, if $l(v) = \mathit{down}$, we apply $\textsc{Reverse}(G_{b'})$, otherwise we can do both. We will show later that using these marks, the increase of the width when drawing $G_{b'}$ "inside" $G_b$ is at most $n' - 3$ instead of $n' - 2$, when $l(v) = \mathit{up}$ or *down* ($b$ the parent-node of $b'$ in $T$).

This method is applied to all 4-connected components of $G$. After directing the edges, this yields a directed acyclic graph, and applying a topological ordering yields an $st$-numbering of the vertices. Applying the algorithm $\textsc{Visibility}(G)$ now gives

the entire drawing. The complete algorithm can now be described more precisely as follows:

COMPACTVISIBILITY (input: a planar graph $G$)
  triangulate $G$;
  construct the 4-block tree $T$ of $G$, and root $T$ at arbitrary node $b$;
  compute the canonical 4-ordering for $G_b$ and direct the edges of $G_b$;
  let $n' = |V(G_b)|$; $l(v_2) = down$, $l(v_{n'-1}) := up$;
  $l(v_i) := unmarked$ for all $v_i \in G_b, i \neq 2, i \neq n'-1$;
  **for** every child $b'$ of $b$ **do** DRAWCOMPONENT($G_{b'}$) **rof**;
  compute an $st$-numbering in the directed graph $G$;
  apply VISIBILITY to $G$;
END COMPACTVISIBILITY

**procedure** DRAWCOMPONENT($G'$);
  **begin**
    **Case** $l(c(G'))$ **of**
      $unmarked$ : NORMAL($G'$); $l(c(G')) := down$; $l(v_{|V(G')|-1}) := up$;
      $up$ : NORMAL($G'$); $l(c(G')) := unmarked$; $l(v_{|V(G')|-1}) := up$;
      $down$ : REVERSE($G'$); $l(c(G')) := unmarked$; $l(v_2) := down$;
    **for** every $v_i \in G'$ with $2 < i < |V(G')| - 1$ **do** set $l(v_i) := unmarked$ **rof**;
    direct the edges of $G'$ $v_i \to v_j$ iff $j > i$;
    **for** every child $b''$ of current node $b'$ in $T$ **do** DRAWCOMPONENT($G_{b''}$) **rof**;
  **end**;

Since $G$ is triangulated, the following lemma can easily be verified:

**Lemma 3.1** *Let $G$ be a triangular planar graph, and let $u = rightvertex(v)$ and $w = rightup(v)$. Setting $x(u,v) := x(v,w) := \max\{x(u,v), x(v,w)\}$ does not increase the width of , $(G)$.*

Let now $b'$ be a (non-root) node in $T$ with parent $b$ in $T$. Let $G'$ be the subgraph of $G$, consisting of all visited 4-connected components in COMPACTVISIBILITY. Let $u, v, w$ be the outerface of $G_{b'}$, with $u \to v$ and $v \to w$ in $G_b$. The following lemma follows ($n' = |V(G_{b'})|$) :

**Lemma 3.2** *If $x(u,w) < x(u,v) < x(v,w)$ in , $(G')$, then applying* NORMAL($G_{b'}$) *has the result that the width of , $(G' \cup G_{b'})$ is at most the width of , $(G') + n' - 3$.*

**Proof:** In , $(G_{b'})$, $x(u,v) - x(u,w) \leq n' - 2$ and $x(v,w) - x(u,w) \leq n' - 1$ by Theorem 2.3. In , $(G')$, $x(u,v) - x(u,w) \geq 1$ and $x(v,w) - x(u,w) \geq 2$. Hence the width of , $(G' \cup G_{b'})$ is at most the width of , $(G') + (n'-1) - 2$. □

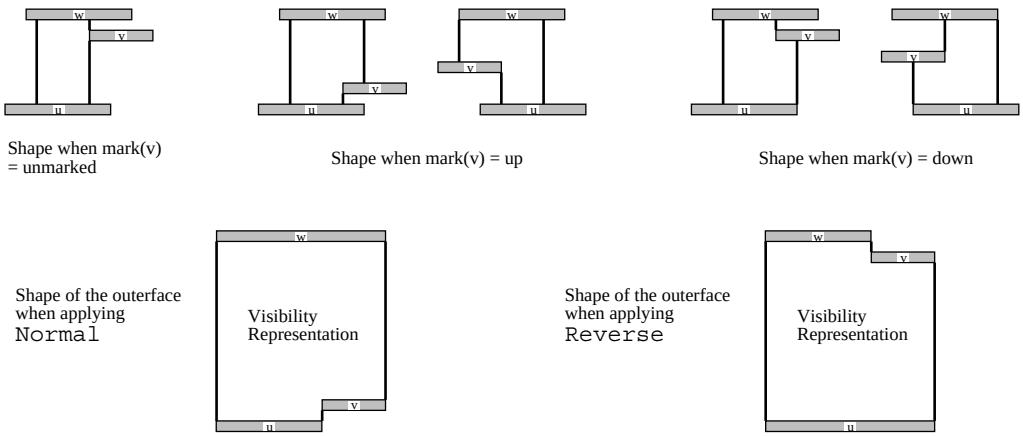



Figure 5: The shape of the faces with respect to $l(v)$, NORMAL($G_{b'}$) and REVERSE($G_{b'}$).

The same can be proved for $x(v,w) < x(u,v) < x(u,w)$ and applying NORMAL, or when $x(u,w) < x(v,w) < x(u,v)$ or $x(u,v) < x(v,w) < x(u,w)$ and applying REVERSE. See Figure 5 for an illustration of this. Assume now that $G'$, $b$ and $b'$ are defined as in the previous lemma. The following lemma can now be proved. Let $v = c(G_{b'})$.

**Lemma 3.3** *If $v$ is marked, then after applying* NORMAL($G_{b'}$) *if $l(v) = up$ and* REVERSE($G_{b'}$) *if $l(v) = down$, the width of , $(G' \cup G_{b'})$ is at most the width of , $(G') + n' - 3$,*

**Proof:** Let $u, v, w$ be the separating triplet of $G_{b'}$, with $u \to v$ and $v \to w$ in $G_b$, thus $v = c(G_{b'})$. If $out(v) = 1$ in $G_b$, then $l(v) = up$. Hence either $x(v,w) < x(u,v) < x(u,w)$ in , $(G')$ or we can change , $(G')$ without increasing the width (by Lemma 3.1) such that $x(u,w) < x(u,v) < x(v,w)$ in , $(G')$. Applying Lemma 3.2 yields the desired result. The same follows when $in(v) = 1$ in $G_b$, thus when $l(v) = down$.

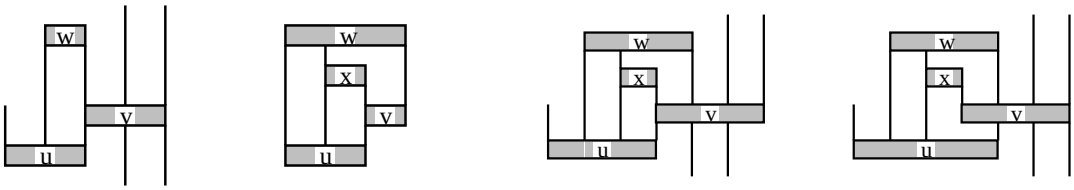
The remaining case is when $v$ was $c(G_{b''})$ for some $b'' \neq b'$, and at the moment of visiting $b''$, $l(v) = unmarked$. Let the separating triplet of $G_{b''}$ be $u', v, w'$, with $u' \to v$ and $v \to w'$. Let $G''$ be the subgraph of $G$, consisting of the visited 4-connected components at the moment of visiting $G_{b''}$. By Lemma 3.1 we may assume that $x(u',v) = x(v,w')$ in , $(G'')$.

Consider the case $x(u',w') < x(u',v)$ (the case $x(u',v) < x(u',w')$ goes similar). In COMPACTVISBILITY NORMAL($G_{b''}$) is applied. Since $in(v) = 1$ in $G_{b''}$, we can set $x(u',v)$ to $x(v,w')$ in , $(G_{b''})$ without increasing the width (see Lemma 3.1). This has the result that $x(v,leftup(v)) < x(u',v)$ in , $(G')$. If $w = leftup(v)$ then the proof

is completed by observing that $x(u,w) < x(v,w) < x(u,v)$ in , $(G')$ and applying Lemma 3.2.

If $w \neq \mathit{leftup}(v)$, then $w = \mathit{rightup}(v)$ and $u = \mathit{rightvertex}(v)$. By Lemma 3.1 we may assume that both $x(u',v) = x(v,w')$ and $x(\mathit{rightvertex}(v),v) = x(v,\mathit{rightup}(v))$ holds in , $(G'')$. But since $x(u',v) < x(v,w')$ holds in , $(G_{b''})$ it directly follows that $x(\mathit{rightvertex}(v),v) < x(v,\mathit{rightup}(v))$ in , $(G'' \cup G_{b''})$. Again the proof is completed by observing that $x(u,v) < x(v,w) < x(u,w)$ in , $(G')$ and applying Lemma 3.2. □

See Figure 6 for an illustration of the proof of Lemma 3.3.




(a) The face and the 4-connected component.

(b) The 2 different possibilities for drawing inside the face.

Figure 6: Illustration of Lemma 3.3.

**Lemma 3.4** *The width of the visibility representation of $G$ is at most $\lfloor \frac{3}{2}n \rfloor - 3$.*

**Proof:** Let $b_1, \ldots, b_p$ be the nodes of $T$ in visiting order. Let $K_i$ be the number of marked vertices after visiting $b_i$ $(1 \leq i \leq p)$. Let $K_0$ be the initial number of marked vertices. $K_0 = 2$, since initially only $v_2$ and $v_{n'-1}$ are marked. When we visit $G_{b_i}$, then vertex $v_2$ or $v_{n'-1}$ is added to the current graph and is unmarkded. If $l(c(G_{b_i})) \neq \mathit{unmarked}$, then the increase in width is at most $|V(G_{b_i})|-3$ and $l(c(G_{b_i}))$ becomes *unmarked*, i.e., $K_i = K_{i-1}$. If $l(c(G_{b_i})) = \mathit{unmarked}$, then the increase in width is at most $|V(G_{b_i})|-2$ and $l(c(G_{b'}))$ becomes *down*, i.e, $K_i = K_{i-1}+2$. Hence in both cases when visiting $G_{b_i}$, the width of the drawing increases by at most $|V(G_{b_i})| - 3 + \frac{K_i - K_{i-1}}{2}$. $|V(G_{b'}| - 3$ is also precisely the number of added vertices of $G_{b'}$.

Since the width of , $(G_{b_1})$ is at most $|V(G_{b_1})|-1$ and $K_p$ is even and $K_p \leq n-2$ (the source and the sink of $G$ never get marked), it follows that the total width of , $(G)$ is at most $n - 1 + \lfloor \frac{K_p - K_0}{2} \rfloor = n - 1 + \lfloor \frac{n-2-2}{2} \rfloor \leq \lfloor \frac{3}{2}n \rfloor - 3$. □

Regarding the time complexity we show in Section 4 that the 4-block tree can be computed in linear time. Computing a canonical 4-ordering also requires linear time [13]. We maintain the direction of the edges of the visited 4-connected components, and from this $c(G_{b'})$ can be determined directly in $O(1)$ time. Finally VISIBILITY$(G)$ is applied, which requires linear time [18, 19]. This completes the following theorem.

**Theorem 3.5** *There is a linear time and space algorithm for computing a visibility representation of a planar graph $G$ on a grid of size at most $(\lfloor \frac{3}{2}n \rfloor - 3) \times (n-1)$.*

Consider the graph of Figure 1. In Figure 4 the 4-block tree is given. The visibility representation of the root-block is given in Figure 3. Drawing the other 4-connected components inside and applying VISIBILITY leads to the drawing as given in Figure 1. Notice that $l(D), l(G)$ and $l(I)$ are *down*, $l(F), l(J)$ and $l(K)$ are *up*, all the other vertices of the graph are unmarked. Hence 6 vertices are marked, and the total width is at most $n - 1 + \frac{6}{2} = 15$. (The width in the drawing in Figure 1 is 12.)

# 4 Constructing the 4-block tree

In this section we show a method for constructing the 4-block tree of a triangulated planar graph. Since this class of graphs has some special properties, there is no need to use the complicated algorithm of Kanevsky et al., which builds a 4-block tree of a general graph in $O(n \cdot \alpha(m,n) + m)$ time [9]. In our case a separating triplet is a separating triangle, which forms the basis for the algorithm.

For determining the separating triangles, we use the algorithm of Chiba & Nishizeki [2] for determining triangles in a graph. (In [17], Richards describes another linear time algorithm.) Chiba & Nishizeki first sort the vertices in $v_1, \ldots, v_n$ in such a way that $deg(v_1) \geq deg(v_2) \geq \ldots \geq deg(v_n)$. Observing that each triangle containing vertex $v_i$ corresponds to an edge joining two neighbors of $v_i$, they first mark all vertices $u$ adjacent to $v_i$ (for the current index $i$). For each marked $u$ and each vertex $w$ adjacent to $u$ they test whether $w$ is marked. If so, a triangle $v_i, u, w$ is listed. After this test is completed for each marked vertex $u$, they delete $v_i$ from $G$ and repeat the procedure with $v_{i+1}$. Starting with $v_1$, this algorithm lists all triangles without duplication in $n-2$ steps.

In our case, we are looking for *separating* triangles. If a triangle is not separating, then it is a face in the triangulated planar graph. To test this, we store the embedding of the original graph also in adjacency lists, say in adjacency lists $Adj(v)$ for all $v \in G$. If a triangle $u, v, w$ is not separating, (i.e., a face), then $u$ and $w$ appear consecutively in $Adj(v)$, which can be tested in $O(1)$ time by maintaining crosspointers. To compute the time complexity of this algorithm, Chiba & Nishizeki use the *arboricity* of $G$, defined as the minimum number of edge-disjoint forests into which $G$ can be decomposed, and denoted by $a(G)$ [7].

**Lemma 4.1 ([2])** $\sum_{(u,v)\in E} \min\{deg(u), deg(v)\} \leq 2 \cdot a(G) \cdot m$.

Using this lemma Chiba & Nishizeki show that the time complexity of the algorithm is $O(a(G) \cdot m)$. If $G$ is planar then $a(G) \leq 3$ ([7]), so the algorithm runs in $O(n)$ time in case $G$ is planar.

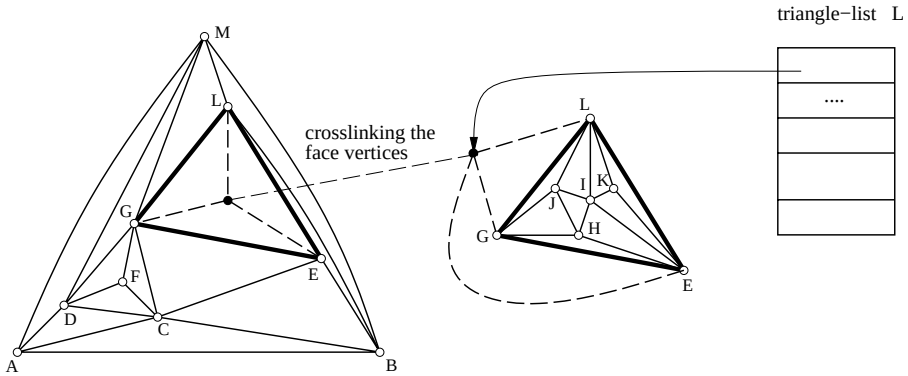



Figure 7: The data structure for constructing the 4-block tree.

To obtain the 4-block tree we introduce now the following data structure: Let $L$ be the list of separating triangles. $L(u,v,w)$ denotes the record in $L$, containing separating triangle $u,v,w$. $L(u,v,w)$ contains the edges $(u,v),(v,w)$ and $(w,u)$, and there are crosspointers between $L$ and the edges and vertices in $G$.

We now want to "sort" the separating triangles, containing edge $(u,v)$. Hereto we do the following: Let $Adj(v) = w_0, w_1, \ldots, w_{d-1}$ (in clockwise order around $v$ with respect to a planar embedding). We sort the separating triangles $v, w_i, w_j$ stored at $v$ in order with respect to $w_0, \ldots, w_{d-1}$. If there are separating triangles $v, w_i, w_j$ and $v, w_i, w_k$ then we set a pointer from edge $(v, w_i)$ in $L(v, w_i, w_j)$ to edge $(v, w_i)$ in $L(v, w_i, w_k)$, if $k > j$ (addition modulo $d$). We do this for every vertex $v \in G$. Of course, when visiting vertex $w_i$ and considering separating triangles $v, w_i, w_j$ and $v, w_i, w_k$, there is no need to place another directed edge between $L(v, w_i, w_j)$ and $L(v, w_i, w_k)$.

Observe now that when edge $(u,v)$ in $L(u,v,w)$ has no outgoing edge, then this means that when we split $G$ at $(u,v),(v,w),(w,u)$ into two subgraphs, say $G_1$ and $G_2$, then all other separating triangles, containing $(u,v)$, belong to either $G_1$ or $G_2$. We start at an arbitrary separating triangle in $L$, say $u,v,w$, where each edge in $L(u,v,w)$ has either an incoming or an outgoing edge. We split the graph at $(u,v),(v,w)$ and $(w,u)$ into two subgraphs, say $G_1$ and $G_2$. Let $Adj(v) = w_0, \ldots, w_{d-1}$ and let $u = w_0$ and $w = w_i$, $0 < i < d$. To obtain $G_1$ and $G_2$, we split $Adj(v)$ into two adjacency lists, say $Adj_1(v)$ and $Adj_2(v)$ with $Adj_1(v) = w_0, \ldots, w_i$ and $Adj_2(v) = w_i, \ldots, w_{d-1}, w_0$ (similar for $Adj(u)$ and $Adj(w)$). Let $Adj_1(v)$ correspond to $G_1$. If all other separating triangles, containing $(u,v)$, belong to $G_1$, then we introduce $w_0$ in $Adj_2(v)$. This yields that all other separating triangles in $L$, containing $(u,v)$, still point to the right edge in the data structure, viz. in the adjacency list of $G_1$. Testing whether the other separating triangles, containing $(u,v)$, belong to $G_1$ or $G_2$ can be tested by checking whether $(u,v)$ in $L(u,v,w)$ has an incoming or an outgoing edge. we introduce $w_0$ in $Adj_1(v)$. Similar is done for

the edges $(v, w)$ and $(w, u)$ with respect to $Adj(w)$ and $Adj(u)$. We mark $L(u, v, w)$ as visited and delete the incoming and outgoing edges of $L(u, v, w)$, and we continue until all separating triangles in $L$ are visited.

To construct the 4-block tree, we apply a simple traversal through the datastructure for determining the connected components. The connections via the face vertices give the connections in the 4-block tree. For every 4-connected component we add pointers to its three vertices on the outerface. The complete algorithm can now be described as follows:

CONSTRUCT 4-BLOCK TREE
  enumerate all separating triangles and store them in $L$;
  sort the separating triangles and add directed edges in $L$;
  **while** not every triangle in $L$ is visited **do**
    Let $L(u, v, w)$ be a record in $L$ such that each edge $(u, v), (v, w)$ and $(w, u)$ in $L(u, v, w)$ has either an incoming or outgoing edge;
    split the graph at edges $(u, v), (u, w), (v, w)$;
    set a pointer between the two corresponding faces;
  **od**;
  determine the connected components and construct the 4-block tree;
END CONSTRUCT 4-BLOCK TREE

**Theorem 4.2** *The 4-block tree of a triangulated planar graph can be constructed in linear time and space.*

**Proof:** Determining and storing the separating cycles requires $O(n)$ time, because every planar graph has at most $n - 4$ separating triangles. Sorting the separating triangles at vertex $v$ can be done in $O(deg(v))$ time by using (double) bucket-sort, since vertex $v$ belongs to at most $deg(v) - 2$ separating triangles. Hence the total sorting time is $O(n)$. By maintaining a sublist of $L$ where we store all separating triangles, which can be visited next, we can find the next separating triangle in $O(1)$ time. Since there are crosspointers between the edges and vertices in $G$ and the corresponding entry in $L$, we can split the graph at the separating triangle in $O(1)$ time. Determining the connected components and building up the 4-block tree is achieved by a simple traversal through the graph, which completes the proof. □

# 5 Final Remarks

In this paper a new and rather simple method for constructing a visibility representation is given, based on the canonical ordering for 4-connected triangular planar graphs, the 4-block tree, and the general algorithm for constructing a visibility representation, introduced by Rosenstiehl & Tarjan [18] and Tamassia & Tollis [19].

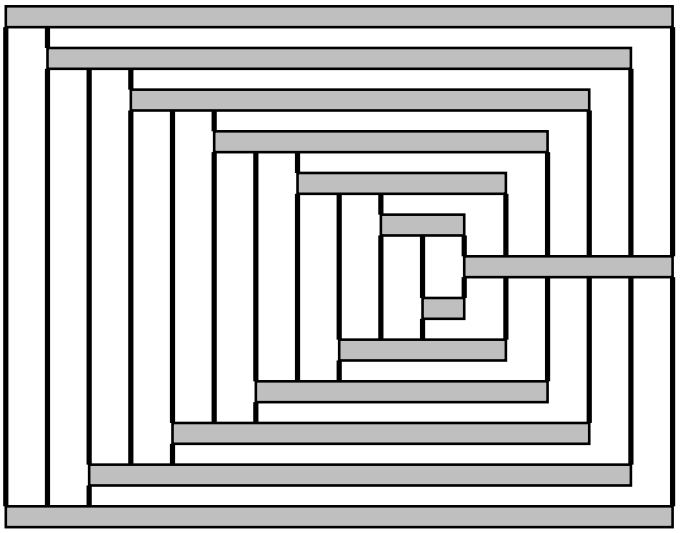


Figure 8: A graph requiring an $(\lfloor \frac{3}{2}n \rfloor - 3) \times (n-1)$ grid by CompactVisibility.

We decreased the width from $2n-5$ to $\lfloor \frac{3}{2}n \rfloor - 3$. This bound is tight, since there exist planar graphs, for which the algorithm CompactVisibility indeed requires an $(\lfloor \frac{3}{2}n \rfloor - 3) \times (n-1)$ grid (see Figure 8). For 4-connected planar graphs, it was already known that a visibility representation can be constructed on an $(n-1) \times (n-1)$ grid [13].

The important open question is whether the bound of $\lfloor \frac{3}{2}n \rfloor - 3$ for the width is also a lower bound, or in other words: do there exist planar graphs with $n$ vertices, for which any visibility representation requires a grid with width at least $\frac{3}{2}n + O(1)$? This is a hard problem, and it is even difficult to construct a class of planar graphs with $n$ vertices, for which any drawing requires a width of size $c \cdot n$ with $c > 1$.

Another (difficult) technique for optimization is the following: From the algorithm it directly follows that for every face $u, v, w$ in the visibility representation of the visited components so far, not drawn as a rectangle, we can set $l(v) = up$ or *down* (according to the shape of the face), when $u \to v$ and $v \to w$. By the algorithm Visibility it follows that $u$ is the rightvertex of $v$ and $w$ is the rightup of $v$. The problem which arises now is: how can we dynamically maintain the longest paths in the dual graph of $G$, while inserting the 4-connected components?

We also presented an algorithm for the construction of the 4-block tree. A linear time implementation in the case of triangulated planar graphs is presented. When the planar graph is not triangulated, then the complexity of constructing a 4-block tree is still $O(n \cdot \alpha(m,n) + m)$, by applying the general algorithm [9]. The open question is whether determining the 4-connected components of a planar graph can be computed in $O(n)$ time. From this it would not be too difficult to compute the

4-block tree in linear time. The problem is already linear time solvable in case of 2- and 3-connected components (see e.g., Hopcroft & Tarjan [8]), hence solving this open problem yields a nice generalization.

As a last subject we consider the method of triangulating planar graphs. It would be interesting to triangulate $G$ such that it is 4-connected. Indeed, $G$ may have separating triangles, in which case it is not possible. Suppose now $G$ has no separating triangles. (This can be tested in linear time.) Can we triangulate $G$ such that the triangulation is 4-connected? Or in general, can we triangulate $G$ such that no new separating triangles are introduced? Can we construct a very compact visibility representation of $G$ without triangulating $G$? These problems have practical applications for our algorithm for constructing visibility representations, but also for constructing the rectangular duals in [13]. We leave these questions for the interested reader.

# References


[1] Bertolazzi, P., R.F. Cohen, G. Di Battista, R. Tamassia and I.G. Tollis, How to draw a series-parallel graph, in: O. Nurmi and E. Ukkonen (Eds.), *Proc. Scand. Workshop on Algorithm Theory (SWAT'92)*, Lecture Notes in Computer Science 621, Springer-Verlag, 1992, pp. 272–283.

[2] Chiba, N., and T. Nishizeki, Arboricity and subgraph listing algorithms, *SIAM J. Comput.* 14 (1985), pp. 210–223.

[3] Di Battista, G., Eades, P., and R. Tamassia, I.G. Tollis, *Algorithms for Automatic Graph Drawing: An Annotated Bibliography*, Dept. of Comp. Science, Brown Univ., Technical Report, 1993.

[4] Di Battista, G., R. Tamassia and I.G. Tollis, Area requirement and symmetry display in drawing graphs, *Discrete and Comp. Geometry* 7 (1992), pp. 381–401.

[5] Di Battista, G., R. Tamassia and I.G. Tollis, Constrained visibility representations of graphs, *Inform. Process. Letters* 41 (1992), pp. 1–7.

[6] Fraysseix, H. de, J. Pach and R. Pollack, How to draw a planar graph on a grid, *Combinatorica* 10 (1990), pp. 41–51.

[7] Harary, F., *Graph Theory*, Addison-Wesley Publishing Company, Inc., Reading, Mass., 1972.

[8] Hopcroft, J., and R.E. Tarjan, Dividing a graph into triconnected components, *SIAM J. Comput.* 2 (1973), pp. 135–158.

[9] Kanevsky, A., R. Tamassia, G. Di Battista and J. Chen, On-line maintenance of the four-connected components of a graph, in: *Proc. 32th Annual IEEE Symp. on Found. of Comp. Science*, Puerto Rico, 1991, pp. 793–801.

[10] Kant, G., Drawing planar graphs using the *lmc*-ordering, *Proc. 33th Ann. IEEE Symp. on Found. of Comp. Science*, Pittsburgh, 1992, pp. 101–110.

Revised and extended version in:

[11] Kant, G., *Algorithms for Drawing Planar Graphs*, PhD thesis, Dept. of Computer Science, Utrecht University, 1993.

[12] Kant, G., On triangulating planar graphs, submitted to *Information and Computation*, 1993.

[13] Kant, G., and X. He, Two Algorithms for Finding Rectangular Duals of Planar Graphs, Tech. Report RUU-CS-92-41, Dept. of Computer Science, Utrecht University, 1992.

[14] Nummenmaa, J., Constructing compact rectilinear planar layouts using canonical representation of planar graphs, *Theoret. Comp. Science* 99 (1992), pp. 213–230.

[15] Otten, R.H.J.M., and J.G. van Wijk, Graph representation in interactive layout design, in: *Proc. IEEE Int. Symp. on Circuits and Systems*, 1978, pp. 914–918.

[16] Read, R.C., A new method for drawing a graph given the cyclic order of the edges at each vertex, *Congr. Numer.* 56 (1987), pp. 31–44.

[17] Richards, D., Finding short cycles in planar graphs using separators, *J. Alg.* 7 (1986), pp. 382–394.

[18] Rosenstiehl, P., and R.E. Tarjan, Rectilinear planar layouts and bipolar orientations of planar graphs, *Discr. and Comp. Geometry* 1 (1986), pp. 343–353.

[19] Tamassia, R., and I.G. Tollis, A unified approach to visibility representations of planar graphs, *Discr. and Comp. Geometry* 1 (1986), pp. 321–341.

[20] Tamassia, R., and I.G. Tollis, Planar grid embedding in linear time, *IEEE Trans. Circuits and Systems* 36 (1989), pp. 1230–1234.