# Quasi–Orthogonal Drawing of Planar Graphs

Gunnar W. Klau* Petra Mutzel*

Max–Planck–Institut für Informatik
Saarbrücken, Germany

**Abstract**

Orthogonal drawings of graphs are highly accepted in practice. For planar graphs with vertex degree of at most four, Tamassia gives a polynomial time algorithm which computes a region preserving orthogonal grid embedding with the minimum number of bends. However, the graphs arising in practical applications rarely have bounded vertex degree. In order to cope with general planar graphs, we introduce the quasi–orthogonal drawing model. In this model, vertices are drawn on grid points, and edges follow the grid paths except around vertices of high degree. Furthermore we present an extension of Tamassia's algorithm that constructs quasi–orthogonal drawings. We compare the drawings to those obtained using related approaches.

## 1 Introduction

Visualizing discrete structures and relations is becoming more and more important. The field of *Automatic Graph Drawing* provides algorithms to construct geometric representations of the underlying graph structures. Interesting and challenging applications arise in many scientific fields, e.g., software engineering, astrophysics, automation engineering, and circuit layout in VLSI design.

We consider drawings of undirected connected planar graphs and concentrate on the aesthetic criteria orthogonality, a low number of bends, and a small drawing area. Additionally, the vertices should be represented by geometrical objects of equal size. Drawings that fulfill these standards are highly accepted in practice because of their excellent readability. In an orthogonal grid drawing, the vertices have integer coordinates and the edges follow the horizontal and vertical grid lines. Unfortunately these drawings are only admissible for graphs with vertex degree of at most four. For this class of graphs, the algorithm in [Tam87] computes an orthogonal grid drawing with the minimum number of bends preserving the topological structure of the input graph.

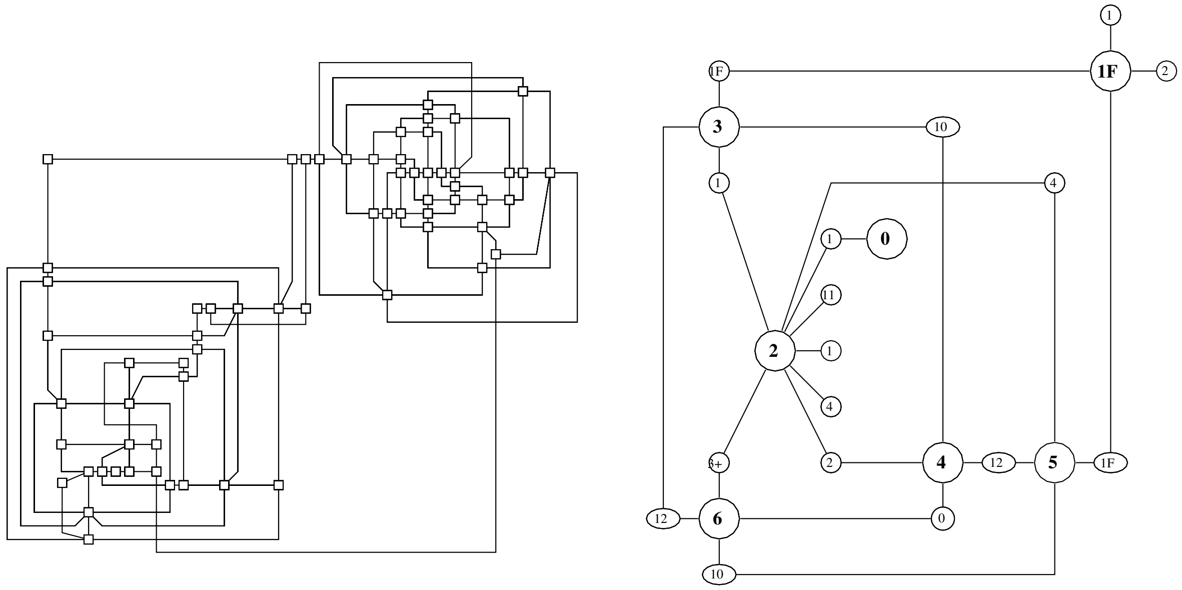
Several extensions have been formulated in order to deal with planar graphs of arbitrary vertex degree. In the *Giotto* model [TDB88], vertices of high degree are drawn as boxes occupying more than one grid point. Another approach is the *Kandinsky* model [FK96] where vertices are placed on a coarse grid and edges are allowed to run on a finer grid. The fineness of the edge grid depends on the maximal vertex degree.

In the approach suggested in this paper, the vertices are represented by objects of equal size and edges run mainly on grid paths. They are allowed to leave the grid only around vertices of high degree; here, the first edge segment may run diagonally through the grid. We formally express this

*E–mail of the authors: guwek@mpi-sb.mpg.de, mutzel@mpi-sb.mpg.de. This work is partially supported by the Bundesministerium für Bildung, Wissenschaft, Forschung und Technologie (No. 03–MU7MP1–4).

model defining the term *quasi–orthogonal grid embedding* and present the algorithm *Quod*, which produces drawings according to this standard.

Figure 1 shows two examples drawn with *Quod*. Note that neither graph is planar; the edge crossings have been created by first computing the maximum planar subgraph [JM96] and subsequent reinsertion of the missing edges by applying a heuristic based on shortest path computations in the dual graph.




(a) (b)

Figure 1: Drawings computed by the *Quod* algorithm. (a) Graph drawing contest 1994, graph B. (b) State diagram from automation engineering.

Sections 2 and 3 provide some fundamental definitions and present the bend minimization algorithm by [Tam87] which forms the basis for our extension. Section 4 introduces the quasi–orthogonal model and describes the algorithm *Quod* that produces drawings for this model. We conclude with Section 5 in which we briefly introduce the models *Giotto* and *Kandinsky* and compare the drawings achieved by the different approaches.

## 2 Preliminaries

An undirected graph is *planar* if it is embeddable in the plane, i.e., if it is possible to map its vertices to distinct points in $\mathbb{R}^2$ and the edges to non crossing Jordan curves connecting their endpoints. Such a mapping is called a *planar geometric embedding* or simply an *embedding* when this causes no confusion. It induces a partition of the plane into a set of regions or *faces* which we will denote by $F$. The unbounded region will be referred to as the *external face* $f_0$, other faces as *internal faces*. Euler's formula correlates the cardinalities of the vertex set $V$, the edge set $E$, and the face set $F$ of a connected planar graph $G$. It states that in any embedding of $G$ we have $|F|-|E|+|V|=2$. An *(orthogonal) grid embedding* is an embedding in which the vertices are mapped to integer grid points and the edges to paths along the grid lines. The topological structure of a planar graph can be described using a *planar* *representation* $P$ that specifies for each face the list of bounding edges in circular order; we define $P$ as a function from the set of faces to lists of edges. An edge appearing twice in one and the same list $P(f)$ is called a *bridge* in $f$. The *degree* of a face $f$ is the number of bounding edges and is denoted as $\delta(f)$, for a vertex $v$ its degree $\delta(v)$ denotes the number of $v$'s neighbours. The maximal vertex degree of a graph $G = (V,E)$ is defined as $\Delta(G) = \max\{\delta(v) \mid v \in V\}$. We call a graph $G$ *k–planar* if it is planar and if $\Delta(G)$ is smaller than or equal to $k$. The deletion of a subset of vertices $U \subseteq V$ together with its adjacent edges from a graph $G = (V,E)$ is written as $G - U$.

A *network* $N$ is a tuple $(U,A,b,l,u,c)$ (see [AMO93]). $U$ and $A$ build the node and arc set of a directed graph. The vector $b \in \mathbb{Z}^U$ contains for each node $i \in U$ its supply (if $b_i \geq 0$) or demand ($b_i < 0$). Vectors $l,u$ and $c \in \mathbb{Z}^A$ denote lower bounds, upper bounds and costs, respectively. The minimum cost flow problem is to find a legal flow vector $\chi \in \mathbb{Z}^A$ that minimizes the total cost. Formally, it is stated as

$$\begin{array}{lll}
\min & \sum\limits_{a \in A} c_a \chi_a & \\
\text{s.t.} & \sum\limits_{\{j|(i,j)\in A\}} \chi_{(i,j)} - \sum\limits_{\{j|(j,i)\in A\}} \chi_{(j,i)} = b_i & \forall i \in U \\
& \qquad l_a \leq \chi_a \leq u_a & \forall a \in A.
\end{array}$$

# 3 Bend Minimization for 4–Planar Graphs

Given a 4–planar graph $G$ with planar representation $P$, the algorithm by Tamassia [Tam87] computes a grid embedding with the minimum number of bends in polynomial time. The computed embedding is *region preserving*, i.e., the underlying topological structure given in $P$ is not changed by the algorithm. The problem of finding the bend minimum grid embedding with respect to every possible planar representation is shown to be NP–hard [GT95].

The bend minimization algorithm runs in several phases (see Figure 2). Each step adds more information to the given topological description. In the minimization phase a network $N$ is constructed depending on $P$ in which each feasible flow corresponds to a possible shape of $G$.

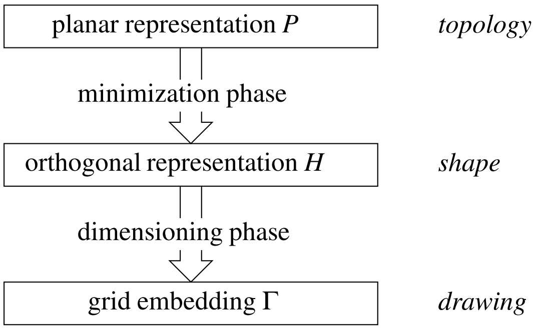
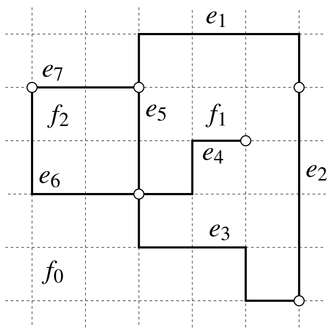



Figure 2: The bend minimizing algorithm.

In particular, the minimum cost flow leads to an embedding with the lowest number of bends since each unit of cost is associated with a bend in the drawing. The flow is used to build a so–called *orthogonal representation* $H$ which describes the shape of the latter drawing in terms of bends occurring along the edges and angles formed by the edges. Formally, $H$ is a function from the set of faces $F$ to lists of triples $r = (e_r, s_r, a_r)$ where $e_r$ is an edge, $s_r$ is a bit string, and $a_r$ is the angle formed with the following edge inside the appropriate face. The bit string $s_r$ provides information about the bends along edge $e_r$, and the $k$th bit describes the $k$th bend on the right side of $e_r$ where a 0 indicates a $90°$ bend and a 1 a $270°$ bend. The empty string $\varepsilon$ is used to characterize straight line edges. Figure 3 shows an example.




$$
\begin{aligned}
H(f_1) &= \big((e_1,00,180),(e_2,\varepsilon,90),(e_3,010,90),\\
&\quad\ (e_4,10,360),(e_4,10,90),(e_5,\varepsilon,180)\big)\\
H(f_2) &= \big((e_6,0,90),(e_7,\varepsilon,90),(e_5,\varepsilon,90)\big)\\
H(f_0) &= \big((e_7,\varepsilon,270),(e_6,1,90),(e_3,101,270),\\
&\quad\ (e_2,\varepsilon,180),(e_1,11,90)\big)
\end{aligned}
$$

Figure 3: Grid embedding and orthogonal representation of a 4–planar graph.

There are four necessary and sufficient conditions for an orthogonal representation $H$ to be a valid shape description of some 4–planar graph:

**P1** There is a 4–planar graph whose planar representation $P$ is identical to that given by $H$ restricted to the $e$–fields. We say that *$H$ extends $P$*.

**P2** Let $r$ and $r'$ be the elements in $H$ with $e_r = e_{r'}$. Since each edge is contained twice in $H$ these pairs always exist. Then string $s_{r'}$ can be obtained by applying bitwise negation to the reversion of $s_r$.

**P3** Let $|s|_0$ and $|s|_1$ denote the numbers of zeroes and ones in string $s$, respectively. Define for each element $r$ in $H$ the value

$$\rho(r) = |s_r|_0 - |s_r|_1 + (2 - \frac{a_r}{90}).$$

Then for each face $f$

$$\sum_{r\in H(f)} \rho(r) = \begin{cases} +4 & \text{if } f \text{ is an internal face} \\ -4 & \text{if } f \text{ is the external face } f_0. \end{cases}$$

**P4** For each vertex $v \in V$ is

$$\sum_{e_r=(u,v)} a_r = 360 \qquad \forall u \in V,$$

i.e., the angles around $v$ given by the $a$–fields sum up to $360°$.

We say that a drawing $\Gamma$ *realizes* $H$ if $H$ is a valid description for the shape of $\Gamma$. Figure 3 shows an orthogonal representation $H$ and a grid embedding realizing $H$. Note that the number of bends in any drawing that realizes $H$ is

$$b(H) = \frac{1}{2} \sum_{f\in F} \sum_{r\in H(f)} |s_r|.$$

Let $G=(V,E)$ be the input graph with planar representation $P$ defining the face set $F$. For the cardinalities $|V|$ and $|E|$ we write $n$ and $m$, respectively. The construction of the underlying network $N$ follows [GT97]: Let $U=U_F\cup U_V$ denote its node set. Then for each face $f\in F$ there is a node in $U_F$ and for each vertex $v\in V$ there is one in $U_V$. Nodes $u_v\in U_V$ supply $b(u_v)=4$ units of flow and nodes $u_f\in U_F$ consume

$$-b(u_f)=\begin{cases}2\delta(f)-4 & \text{if } f \text{ is an internal face}\\ 2\delta(f)+4 & \text{if } f \text{ is the external face } f_0\end{cases}$$

units of flow. Thus, the total supply is $4n$ and the total demand is

$$\sum_{f\neq f_0}(2\delta(f)-4)+2\delta(f_0)+4=2\sum_f\delta(f)-4|F|+8=4m-4|F|+8$$

which is equal to the total supply, according to Euler's formula. The arc set $A$ of network $N$ consists of two sets $A_V$ and $A_F$ where

$$\begin{aligned}A_V &= \{(u_v,u_f)\mid u_v\in U_V, u_f\in U_F, v \text{ is adjacent to } f\} \quad \text{and}\\ A_F &= \{(u_f,u_g)\mid u_f\neq u_g\in U_F,\ f \text{ is adjacent to } g\}\\ &\cup\ \{(u_f,u_f)\mid f \text{ contains a bridge}\}.\end{aligned}$$

Arcs in $A_V$ have lower bound 1, capacity 4, and cost 0. Each unit of flow represents an angle of 90°, so a flow in an arc $(u_v,u_f)\in A_V$ corresponds to the angles formed at vertex $v$ inside face $f$. Note that there can be more than one angle, see for example Figure 3 where the vertex common to edges $e_6,e_5,e_4$, and $e_3$ builds two angles in $f_1$. Precisely, the flow in $(u_v,u_f)$ corresponds to the sum of the angles at $v$ inside $f$. Following this interpretation, flow in arcs $(u_f,u_g)\in A_F$ find their analogy in bends occurring along edges separating $f$ and $g$ that form a 90° angle in $f$. Naturally their lower bound is 0, their capacity unbounded, and they have unit cost.

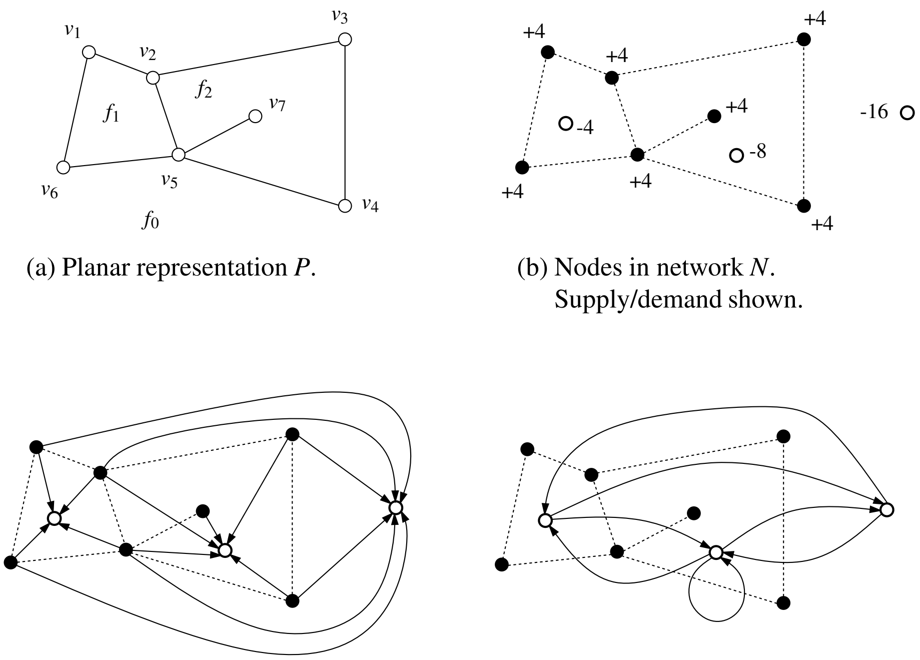
The conservation rule at nodes $u_v\in U_V$ expresses the fact that the angle sum around the corresponding vertex $v$ equals 360°. The units of flow are consumed by the nodes in $U_F$; here, the conservation rule states that every face has the shape of a rectilinear polygon. A planar graph and the transformation into a network is shown in Figure 4 (a) – (d).

There is always a feasible flow in network $N$: The flow produced by nodes in $U_V$ can be transported to nodes in $U_F$ where it satisfies the demand. If it is not possible to satisfy every node in $U_F$ by exclusively using arcs $A_V$, units of flow can be shifted without restriction between nodes in $U_F$ because of their mutual interconnection by arcs in $A_F$. Every feasible flow can be used to construct an orthogonal representation for the input graph $G$, in particular the minimum cost flow, leading to the orthogonal representation with the minimum number of bends. The following lemma states the analogy between flows in the network and orthogonal representations.

**Lemma 1.** *[Tam87] Let $G$ be the input graph, $P$ its planar representation, and $N$ the constructed network. For each integer flow $\chi$ in network $N$, there is an orthogonal representation $H$ that extends $P$ and whose number of bends is equal to the cost of $\chi$. The flow $\chi$ can be used to construct the orthogonal representation.*

*Proof.* We extend $P$ using flow $\chi$ such that the resulting orthogonal representation $H$ fulfills properties (P1) – (P4). We then show that the number of bends in $H$ equals the cost of flow $\chi$. Let $R(v,f)$ be the subset of $H(f)$ with endpoints of entry $e_r$ equal to $v$. These are the entries whose angle values are determined by $\chi_{(u_v,u_f)}$. Formally,

$$R(v,f)=\{r\in H(f)\mid e_r=(u,v) \text{ for some } u\in V\}.$$




(a) Planar representation $P$.

(b) Nodes in network $N$. Supply/demand shown.

(c) Arcs in $A_V$. Capacity 4, lower bound 1, cost 0.

(d) Arcs in $A_F$. Capacity $\infty$, lower bound 0, cost 1.

Figure 4: Network construction.

Let $r_1, \ldots, r_k$ be the elements of $R(v, f)$ in the order they appear in $H(f)$. Note that $k$ can be at most 4. Setting the $a$–values is done by analyzing the flow in arcs $A_V$. Let $(u_v, u_f)$ be such an arc with flow $\chi_{(u_v,u_f)}$. Then

$$a_{r_1} = (\chi_{(u_v,u_f)} - k + 1) \cdot 90^\circ \quad \text{and} \quad a_{r_i} = 90^\circ \text{ for } 2 \leq i \leq k.$$

In this manner we distribute the units of flow among the angles, guaranteeing that every angle gets at least one unit of flow. Flow in arcs $(u_f, u_g) \in A_F$ determines the number of bends along the edges separating the faces $f$ and $g$. Let $R(f, g) = r'_1, \ldots, r'_k$ denote the elements common to $H(f)$ and $H(g)$ in the order of $H(f)$. For each $r'_i$ let $r''_i$ be the element of $H(g)$ with identical edge entry. Then

$$s_{r'_1} = \begin{cases} 0^{\chi_{(u_f,u_g)}} & \text{if } f = g \\ 0^{\chi_{(u_f,u_g)}} 1^{\chi_{(u_g,u_f)}} & \text{if } f \neq g \end{cases}$$

$$s_{r''_1} = \begin{cases} 1^{\chi_{(u_g,u_f)}} & \text{if } f = g \\ 0^{\chi_{(u_g,u_f)}} 1^{\chi_{(u_f,u_g)}} & \text{if } f \neq g \end{cases}$$

$$s_{r'_i} = s_{r''_i} = \varepsilon \quad \text{for } 2 \leq i \leq k.$$

Property (P1) is not changed and thus valid; (P2) follows directly from the way the bit strings are set. Finally, the flow conservation in the network implies properties (P3) and (P4).

The number of bends in $H$ is

$$b(H) = \frac{1}{2} \sum_f \sum_{r \in H(f)} |s_r| = \frac{1}{2} \sum_{(u_f,u_g) \in A_F} (\chi_{(u_f,u_g)} + \chi_{(u_g,u_f)}) = \sum_{a \in A} c_a.$$

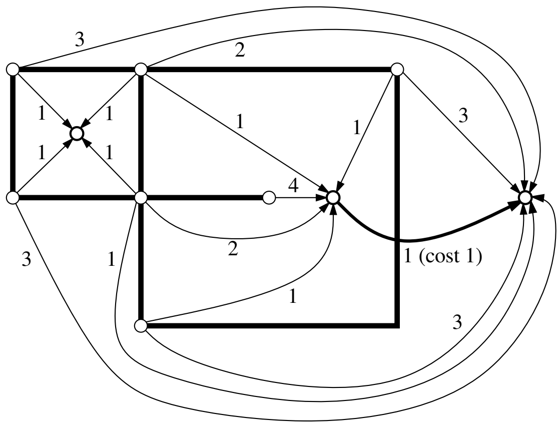



Figure 5: Minimal cost flow in network $N$ and a resulting grid embedding.

Note that in a minimum cost flow solution, we do not have to distinguish between different cases; all bend patterns will be either $0^*$ or $1^*$. □

Figure 5 completes the example from Figure 4, showing the minimum cost flow in the constructed network and a realizing grid embedding for the derived orthogonal representation.

Vice versa, it can be shown that the number of bends in each orthogonal grid embedding of a graph with planar representation $P$ is equal to the cost of some feasible flow in network $N$. This result and Lemma 1 lead to the following theorem, combining the basic results of [Tam87] and [GT97]:

**Theorem 3.1.** *[Tam87, GT97] Let $P$ be a planar representation of a connected 4–planar graph $G$ with $n$ vertices and let $N$ be the appropriate network. Each feasible flow $\chi$ in $N$ corresponds to an orthogonal representation for $G$ that extends $P$ and whose number of bends is equal to the cost of $\chi$. In particular, the minimum cost flow can be used to construct the bend optimal orthogonal representation preserving $P$. This can be done in time $O(n^{7/4}\sqrt{\log n})$.*

At this point, the shape of the latter drawing is determined but the actual coordinates of vertices and bends still need to be computed. The problem is to find a grid embedding $\Gamma$ that realizes the orthogonal representation $H$.

We briefly present the method proposed in [Tam87]: Each face is dissected in rectangular subfaces by eventually adding virtual vertices and edges. For the resulting special kind of orthogonal representation $H'$ it is easy to do the length assignment. The total edge length can be minimized by constructing two separate networks for the horizontal and vertical edge segments, respectively. A unit of flow corresponds to a length unit; the flow conservation expresses the fact that the two opposite sides of each face must be of the same length. Another method computes longest paths in two so–called polar graphs in linear time. The result is an orthogonal grid embedding with minimum height and width for the transformed representation $H'$ — but not necessarily minimum edge length.

After having deleted the virtual vertices and edges, the area consumption of the drawings is $O((n+b)^2)$ where $n$ denotes the number of vertices and $b$ the number of bends in $H$. Though these drawings are optimal for $H'$, they are not for the original shape description $H$. One dimensional compaction strategies known from VLSI design may be applied to improve the area consumption; minimizing the total edge length for arbitrary orthogonal representations is conjectured to be NP–hard.

# 4 The Quasi–Orthogonal Model

The algorithm presented in Section 3 fulfills the desired aesthetic criteria and produces the best results in comparison with other methods used for orthogonal graph drawing [BGL+97]. Unfortunately, it is restricted to the relatively small class of 4–planar graphs; however, the graphs arising in practical applications rarely have bounded vertex degree. Thus, several extensions have been formulated in order to cope with arbitrary planar graphs. In this section we introduce the *quasi–orthogonal* drawing model. Furthermore we present an extension of Tamassia's algorithm that constructs quasi–orthogonal drawings.

In our model, vertices are represented by grid points. This implies that we can no longer stick to orthogonal grid embeddings. Neither do we want to use different grids for vertices and edges. These two requirements enforce some edges to leave the grid lines. We observe that 4–planar subgraphs can still be drawn with the pure orthogonal standard. Our solution is to allow the first segment of any edge leaving a vertex with degree greater than four to run diagonally through the grid. The following definition provides a formal description of this drawing standard:

**Definition 4.1.** A *quasi–orthogonal grid embedding* of a planar graph $G = (V,E)$ is a function $\Gamma$ that maps $V$ to points in the grid and $E$ to sequences of segments whose endpoints lie on the grid. The following properties hold:

(Q1) $\Gamma(v) \neq \Gamma(w)$ for $v,w \in V, v \neq w$.

(Q2) The endpoints of $\Gamma(e)$ are $\Gamma(v)$ and $\Gamma(w)$ for all $e = (v,w) \in E$.

(Q3) For two different edges $e_1$ and $e_2$ the paths $\Gamma(e_1)$ and $\Gamma(e_2)$ do not intersect except possibly at their endpoints.

(Q4) $\Gamma(G - \{v \in V \mid \delta(v) > 4\})$ is an orthogonal grid embedding.

Note that property (Q4) of Definition 4.1 yields a pure orthogonal grid embedding for 4–planar graphs. This implies that every orthogonal grid embedding is also a quasi–orthogonal grid embedding. Figure 6 shows an example for a drawing respecting the properties of Definition 4.1.

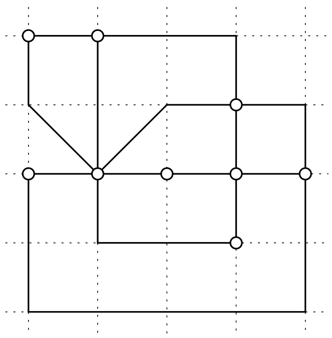


Figure 6: Quasi–orthogonal grid embedding of a 6–planar graph with 7 bends.

In the following we describe the algorithm *Quod* that computes quasi–orthogonal grid embeddings for planar graphs. Related work is discussed in Section 5. In *Quod* high degree vertices $v$ are replaced by faces $f_v$ with $\delta(f_v) = \delta(v)$. The vertices on the boundary of such a newly created representative face $f_v$ correspond to the adjacencies of the former vertex $v$, reflecting the order of the neighbours.

We call these special faces *cages*, in a later phase every high degree vertex will be placed in its corresponding cage. Unlike the method used for the *Giotto* model (which will be briefly discussed in the next section), we do not prescribe on which side of $f_v$ the edges adjacent to $v$ have to leave. We refer to the transformation as $T_1$. It works in the following way (see also Figure 7):

**Transformation $T_1$**

Input: Planar graph $G = (V, E)$, vertex $v \in V$.
Output: $G$, where $v$ is replaced by a cage.

```
forall edges (v,w) ∈ E
  split edge (v,w);
forall faces f adjacent to v
  link pair of new vertices in f by an edge;
delete v;
```

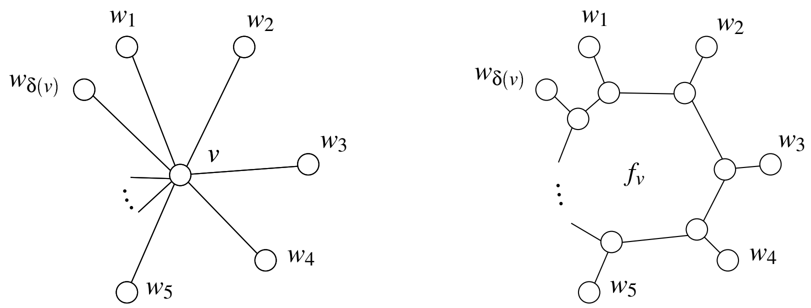



Figure 7: Transformation $T_1$. Replacement of a vertex $v$ with high degree by a cage $f_v$.

After having applied $T_1(G, v)$, the newly created face $f_v$ acts as a representative for the former vertex $v$. Note that $v$ has been replaced by a structure of $\delta(v)$ vertices, each of which has exactly three neighbours. Let $\widetilde{G}$ be the graph that results from applying $T_1$ to every high degree vertex in $G$. In the following analysis, $V_i$ denotes the set $\{v \in V \mid \delta(v) = i\}$. The number of vertices in $\widetilde{G}$ is

$$\begin{aligned} |\widetilde{V}| &= |V| + \sum_{\delta(v)>4} (\delta(v) - 1) \\ &= |V_1| + |V_2| + |V_3| + |V_4| + \sum_{\delta(v)>4} \delta(v) \\ &\leq \sum_i i \cdot |V_i| \leq 2|E| \leq 6n - 12 = O(|V|), \end{aligned}$$

and it is obvious that $\widetilde{G}$ is still planar. Furthermore one can easily verify that the inverse operation of transformation $T_1$ results in the original graph.

Now we can apply the algorithm from Section 3 to $\widetilde{G}$. Since $|\widetilde{V}| = O(|V|)$, the asymptotical notion of running time does not change. But if we do not distinguish between cages and normal faces, we will get a pure orthogonal grid embedding in which the cages have arbitrary rectilinear shape. Since we want to place the formerly high degree vertices in their corresponding cages, it would not be a good idea to let the latter have a too–complicated shape. Therefore we force cages to be of rectangular shape by modifying the network introduced in Section 3. We then show that our modifications indeed achieve the desired results and that our formulation always provides a solution meeting our constraints.

A rectilinear polygon $\Pi$ has the shape of a rectangle if each of the angles inside $\Pi$ does not exceed $180^\circ$. We exploit this fact to formulate a modification of network $N$ concerning the shape of the cages. Consider the set of elements $R = \{r \in H(f) | f \text{ is a cage}\}$. For every element $r \in R$ we must ensure $a_r \leq 180$ and $s_r = 0^*$. These two constraints guarantee that there will be neither a concave angle nor a concave bend in a cage. In [Doo95] a similar method is proposed resulting in drawings according to the *Giotto* standard. In this approach the angles $a_r$ are forced to be equal to $180^\circ$; this leads to an increased total edge length compared to our approach.

Note that the first requirement ($a_r \leq 180^\circ$) is automatically fulfilled. Each vertex $v$ bounding a cage has degree $\delta(v) = 3$ and thus can only form angles of at most $180^\circ$. The second constraint can be formulated by deletion of arcs in $N$. We have to avoid a flow $\chi_{(u_g,u_f)}$ in the case that $f$ represents a cage. Deleting the arc $(u_g,u_f)$ makes such a flow impossible.

We have now modified the network given in Section 3 so that each legal flow in it corresponds to an orthogonal representation with rectangular cages. The following lemma states that these modifications have no influence on the feasibility of the minimum cost flow problem:

**Lemma 2.** *The minimum cost flow in the modified network corresponds to an orthogonal representation with the minimum number of bends under the constraint that every cage has a rectangular shape.*

*Proof.* Let $N$ be the modified network. The modification concerns only the arcs in $A_F$. To prove the lemma we only have to show that the conservation rule at nodes $u_f \in U_F$ still holds. Therefore we consider three cases:

Face $f$ is a cage.

The incoming flow is $\sum_{u_v} \chi_{(u_v,u_f)}$, the outgoing flow is $\sum_{u_g} \chi_{(u_f,u_g)}$. There are exactly four angles of $90^\circ$ in the cage occurring either at vertices or at bends. Thus $|\{u_v \mid \chi_{(u_v,u_f)} = 1\}| + \sum_{u_g} \chi_{(u_f,u_g)} = 4$. We get

$$\begin{aligned}\sum_{u_v} \chi_{(u_v,u_f)} - \sum_{u_g} \chi_{(u_f,u_g)} &= 2\delta(f) - |\{u_v \mid \chi_{(u_v,u_f)} = 1\}| - \sum_{u_g} \chi_{(u_f,u_g)} \\ &= 2\delta(f) - 4 = b_{u_f}.\end{aligned}$$

Face $f$ is the neighbour of a cage.

According to their construction, cages can never be neighbours to other cages, neither can they enclose other faces. For this reason the demand of $u_f$ can be fulfilled by adjacent normal faces.

Face $f$ is neither a cage nor a neighbour of a cage.

In this case there is no difference from the unmodified network. Conservation is guaranteed. □

We now construct an orthogonal embedding $\Gamma$ for $\widetilde{G}$ with one of the compaction methods. During this step we ensure that both the height and the width of a cage measure at least two grid units. This can be easily done in the compaction phase. At this point, we want to reverse the changes of transformation $T_1$. Therefore, we define a second transformation $T_2$ which operates on grid embeddings and places the high degree vertices in their cages. The aim is to minimize the number of bends arising at the boundary of a cage during the process of connecting a high degree vertex $v$ with its adjacent edges. Let $w_1, \ldots, w_{\delta(v)}$ be the vertices on the boundary of the corresponding cage $f_v$ and let $\Gamma(f_v)$

characterize the set of grid points covered by $f_v$. Using straight line edges for the connection of $v$ with its adjacencies, we can save at most four bends. For the detailed and somewhat tedious description of finding the best grid point for $v$, see [Kla97].

**Transformation $T_2$**

Input: Orthogonal grid embedding $\Gamma$, cage $f_v$ with boundary $w_1, \ldots, w_{\delta(v)}$.
Output: $\Gamma$ in which $f_v$ is replaced by the appropriate vertex $v$.

```
place v in Γ(f_v);
/* creating min. number of bends in Γ(f_v) */
for i = 1 to δ(v)
  connect v with w_i;
```

The whole algorithm *Quod* is summarized below. The procedure `tamassia_mod` refers to the modified bend minimizing algorithm described in Section 3 in which each cage is forced to be of rectangular shape. *Quod* has been implemented using $C^{++}$ and LEDA [MN95] and forms a part of the AGD–Library [AGMN97, AGD97].

**The algorithm *Quod***

Input: Planar graph $G = (V, E)$ with planar representation $P$.
Output: Quasi–orthogonal grid embedding $\Gamma$ of $G$.

```
G̃ = G;
while ∃ vertex v ∈ Ṽ with δ(v) > 4
  G̃ = T_1(G̃, v);
Γ̃ = tamassia_mod(G̃, P̃);
Γ = Γ̃;
forall cages f ∈ F̃ if f is cage
  Γ = T_2(Γ, f);
return Γ;
```

# 5 Comparison and Conclusion

Several algorithms and drawing models have been developed to extend Tamassia's algorithm to general planar graphs. In this section we briefly present the approaches *Giotto* and *Kandinsky*. In the discussion following, we compare the different methods.

## 5.1 Giotto

In [TDB88] the following standard is proposed for drawing general planar graphs: Vertices may be represented by rectangular boxes instead of grid points. The input graph is transformed into a 4–planar graph by replacing each vertex $v$ whose degree exceeds 4 by a rectangular structure. The neighbours of $v$ are connected to the four borders of this structure in a prescribed order. This distribution may be computed by a simple heuristic or using the $k$–gonal representation of the input graph. $k$–gonal representations are generalizations of orthogonal representations: For their computation, a similar network can be constructed in which each unit of flow represents an angle of $180/k$ degrees. It is shown that a slightly modified version of Theorem 3.1 still holds.

The basic algorithm is used to compute an embedding for the transformed graph. In order to reflect the prescribed orthogonal representation for the rectangular structures the network must be changed. Note that the drawings produced by this method do not necessarily have the minimum number of bends. Related work appears in [Doo95].

## 5.2 Kandinsky

Another extension of the basis algorithm has been presented in [FK96] and is described in detail in [Föß97]. The model as well as the appropriate algorithm are referred to as *Kandinsky*. In the model, vertices are represented by squares of equal size. In order to draw every edge segment horizontally or vertically, the edge segments may run parallel using a finer grid than that of the vertices. Furthermore the model requires that at least one grid point lies in every face.

For a fixed topology of an input graph the algorithm *Kandinsky* achieves bend minimality in its model. It extends Tamassia's algorithm in the following way:

- The angles $a_r$ can also take the value $0°$. This is used to model parallel edge segments.
- The network is modified in order to consider vertices with degree greater than four.
- The straightforward modification would lead to an unlimited growth of the vertices which is not allowed by the Kandinsky model. An additional network modification is needed that forces exactly one bend for every $0°$ angle. This modification introduces cycles of negative cost. In order to get a useful solution, a special minimum cost flow algorithm which augments the flow exclusively on paths from the source to the sink must be applied.

Drawings produced by the Kandinsky algorithm may be transformed to drawings in the *proportional growth model* [BMT97]. This drawing standard requires vertices to be drawn as boxes whose size is bounded by the corresponding vertex degree. However, the transformed drawings do not have the minimum number of bends in the new model and the drawing area is quite high compared to other approaches.

## 5.3 Comparative discussion

Each of the approaches presented has its advantages and disadvantages. Although the different models make it hard to compare the approaches, we can state some universally valid observations. Figure 8 shows a planar graph drawn with the algorithms *Giotto*, *Kandinsky*, and *Quod*, respectively. Note that the planar representation is the same for each of the three drawings.

In the *Giotto* drawing one can observe that the vertices with degree greater than four may grow independently of their vertex degree. A negative impact of the fixed assignment of edges to the four borders can be seen in the example: The upper edge must leave the box for the high degree vertex at its right side. It is easy to save a bend connecting it to the upper side. The price for distributing adjacent vertices evenly around the boxes is a high number of bends. If we transform the *Quod* output in the *Giotto* standard we get a drawing with only three bends — but unpleasantly long boxes.

Even if the drawing standard of *Quod* is different, the long boxes still exist implicitly in the drawings and lead to very small angles after having replaced the high degree vertices. Both *Giotto*

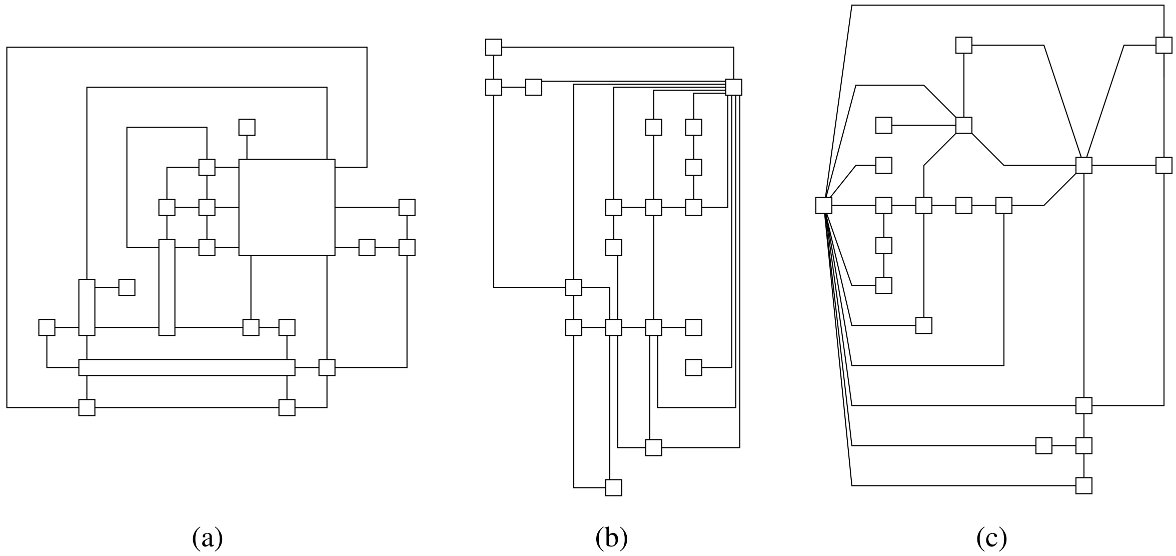



Figure 8: The same planar graph drawn with the different approaches. (a) Giotto: $11 \times 10$ grid, 13 bends. (b) Kandinsky: $7 \times 11$ vertex grid, $12 \times 15$ effective grid, 14 bends. (c) Quod: $10 \times 13$ grid, 18 bends.

and *Quod* suffer from the fact that dimensions play no role during the minimization phase: Bends are avoided by stretching the boxes.

This may not occur in *Kandinsky* drawings. But additional bends is the price paid for keeping the vertices small. For each 0° angle the underlying network structure creates one bend which is necessary in this special model. Often, many edges run parallel on the dense grid (like in the example), in such cases it is not easy to follow the edges. Allowing vertices to grow reasonably leads to drawings in the *proportional growth model* [BMT97]. This can be done by building a new grid taking the grid lines used by the edges. Note that this effective grid is considerably bigger than the primary vertex grid and that the transformed drawings are not bend minimal in the new model.

*Giotto* and *Quod* are very well suited for drawing planarized graphs. These are arbitrary graphs that are planar due to a foregoing planarization step [JM96] and reinsertion of the missing edges. During this process *dummy vertices* are introduced to model the crossings. In a drawing of a planarized graph these vertices have to be drawn on a single grid point $p$, the four adjacent edges have to use the four grid segments around $p$. The algorithms *Giotto* and *Quod* fulfill this requirement automatically. For *Kandinsky* the situation is not as easy: The network needs to be changed again and becomes very complicated.

At present, there is no algorithm producing drawings with the minimum number of bends for the proportional growth model. The drawings could easily be transformed in the quasi–orthogonal drawing standard using the transformation $T_2$ yielding an angular resolution close to the optimum.

Further research should also be invested in minimizing or bounding the size of the drawings. Each of the tested implementations suffer from severe compaction problems. The commonly used method (see Section 3) leads to drawings in which a lot of space is wasted. Compaction heuristics known from VLSI design may lead to improvements; however, the authors are convinced that it is worth designing practically efficient exact compaction algorithms.

# References


[AGD97] AGD–Library – A library of Algorithms for Graph Drawing, 1997. A Software Project supported by the DFG involving groups in Halle, Köln and Saarbrücken, Germany; Available via http://www.mpi-sb.mpg.de/∼mutzel/dfgdraw/dfgdrawsoft.html.

[AGMN97] D. Alberts, C. Gutwenger, P. Mutzel, and S. Näher. AGD–Library: A library of algorithms for graph drawing. In G.F. Italiano and S. Orlando, editors, *WAE '97 (Proc. on the Workshop on Algorithm Engineering)*, Venice, Italy, Sept. 11-13, 1997.

[AMO93] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, Englewood Cliffs, NJ, 1993.

[BGL$^+$97] G. Di Battista, A. Garg, G. Liotta, R. Tamassia, E. Tassinari, and F. Vargiu. An experimental comparison of four graph drawing algorithms. *CGTA: Computational Geometry: Theory and Applications*, 7:303 – 316, 1997.

[BMT97] T. Biedl, B. Madden, and I. Tollis. The three–phase method: A unified approach to orthogonal graph drawing. In G. Di Battista, editor, *Graph Drawing (Proc. GD '97)*, volume 1353 of *Lecture Notes in Computer Science*, pages 391–402. Springer-Verlag, 1997.

[Doo95] M. Doorley. *Automatic Levelling and Layout of Data Flow Diagrams*. PhD thesis, Department of Computer Science and Information Systems, College of Informatics and Electronics, University of Limerick, 1995.

[FK96] U. Fößmeier and M. Kaufmann. Drawing high degree graphs with low bend numbers. In Franz J. Brandenburg, editor, *Graph Drawing (Proc. GD '95)*, volume 1027 of *Lecture Notes in Computer Science*, pages 254–266. Springer-Verlag, 1996.

[Föß97] U. Fößmeier. *Orthogonale Visualisierungstechniken für Graphen*. PhD thesis, Fakultät für Informatik, Eberhard–Karls–Universität zu Tübingen, 1997.

[GT95] A. Garg and R. Tamassia. On the computational complexity of upward and rectilinear planarity testing. In R. Tamassia and I. G. Tollis, editors, *Graph Drawing (Proc. GD '94)*, volume 894 of *Lecture Notes in Computer Science*, pages 286–297. Springer–Verlag, 1995.

[GT97] A. Garg and R. Tamassia. A new minimum cost flow algorithm with applications to graph drawing. In S. North, editor, *Graph Drawing (Proc. GD '96)*, volume 1190. Springer–Verlag, 1997.

[JM96] M. Jünger and P. Mutzel. Maximum planar subgraph and nice embeddings: Practical layout tools. *Algorithmica*, 16(1):33 – 59, 1996. Special Issue on Graph Drawing.

[Kla97] G. W. Klau. Quasi–orthogonales Zeichnen planarer Graphen mit wenigen Knicken. Diplomarbeit, Universität des Saarlandes, Saarbrücken, Fachbereich Informatik, Technische Fakultät, Januar 1997.

[MN95] K. Mehlhorn and S. Näher. LEDA: A platform for combinatorial and geometric computing. *Communications of the ACM*, 38(1):96–102, 1995.

[Tam87] R. Tamassia. On embedding a graph in the grid with the minimum number of bends. *SIAM J. Comput.*, 16(3):421–444, 1987.

[TDB88] R. Tamassia, G. Di Battista, and C. Batini. Automatic graph drawing and readability of diagrams. *IEEE Trans. Syst. Man Cybern.*, SMC-18(1):61–79, 1988.