Computing Orthogonal Drawings with the Minimum Number of Bends

Paola Bertolazzi, Giuseppe Di Battista, *Member*, *IEEE Computer Society*, and Walter Didimo

Abstract—We describe a branch-and-bound algorithm for computing an orthogonal grid drawing with the minimum number of bends of a biconnected planar graph. Such an algorithm is based on an efficient enumeration schema of the embeddings of a planar graph and on several new methods for computing lower bounds of the number of bends. We experiment with such algorithm on a large test suite and compare the results with the state-of-the-art. The experiments show the feasibility of the approach and also its limitations. Further, the experiments show how minimizing the number of bends has positive effects on other quality measures of the effectiveness of the drawing. We also present a new method for dealing with vertices of degree larger than four.

Index Terms—Orthogonal drawings, bends, planar embedding, branch and bound, graph drawing, planar graphs.

1 Introduction

Various graphic standards have been proposed to draw graphs, each one devoted to a specific class of applications. Extensive literature on the subject can be found in [4]. In particular, an *orthogonal drawing* maps each edge into a chain of horizontal and vertical segments and an *orthogonal grid drawing* is an orthogonal drawing such that vertices and bends along the edges have integer coordinates. Orthogonal grid drawings are widely used for graph visualization in many applications, including database systems (entity-relationship diagrams), software engineering (data-flow diagrams), and circuit design (circuit schematics).

Many algorithms for constructing orthogonal grid drawings have been proposed in the literature and implemented into industrial tools. They can be roughly classified according to two main approaches: The *topology-shape-metrics* approach determines the final drawing through an intermediate step in which a planar embedding of the graph is constructed (see, e.g., [27], [25], [28]); the *draw-and-adjust* approach reaches the final drawing by working directly on its geometry (see, e.g., [20], [2], [11], [23]). Since a planar graph has an orthogonal grid drawing if and only if its vertices have degree at most 4, both the approaches assume that vertices have degree at most 4. Such a limitation is usually removed by "expanding" higher degree vertices into two or more vertices [1] or by "shrinking" the distance between edges in the proximity of vertices [12].

In this paper, we adopt the topology-shape-metrics approach. In the topology-shape-metrics approach, the drawing is incrementally specified in three phases. The first phase, *planarization*, determines a planar embedding of the graph, by possibly adding (if the graph is nonplanar) ficticious vertices that represent crossings. The second phase, *orthogonalization*, receives as input a planar embedding and computes an *orthogonal representation*. An orthogonal representation is an equivalence class of orthogonal drawings having the same sequence of bends along the edges and the same angles at the vertices. Roughly speaking, an orthogonal representation is an orthogonal drawing where the length of the segments representing edges is not specified. The third phase, *compaction*, produces the final orthogonal grid drawing trying to minimize the area.

The orthogonalization step is crucial for the effectiveness of the drawing and has been extensively investigated. A very elegant $O(n^2 \log n)$ time algorithm for constructing an orthogonal representation with the minimum number of bends along the edges of an n-vertex embedded planar graph has been presented by Tamassia in [25] and improved in efficiency to $O(N^{7/4} \log N)$ in [15]. It is based on a minimum cost network flow problem that considers bends along edges as units of flow.

However, the algorithm in [25] minimizes the number of bends only within the given planar embedding. Observe that a planar graph can have an exponential number of planar embeddings and it has been shown [6] that the choice of the embedding can deeply affect the number of bends of the drawing. Namely, there exist graphs that, for a certain embedding, have a number of bends that is linear with the number of vertices and, for another embedding, have only a constant number. Unfortunately, the problem of minimizing the number of bends in a variable embedding setting is NP-complete [13], [14]. Optimal polynomial-time algorithms for subclasses of graphs are shown in [6].

Because of the tight interaction between the graph drawing field and applications, the attention to experimental work on graph drawing is rapidly increasing. For example, in [5], an experimental study is presented comparing topology-shape-metrics and draw-and-adjust algorithms for orthogonal grid drawings. The test graphs were generated from a core set of 112 graphs used in

- *P. Bertolazzi is with IASI, CNR, Viale Manzonni, 30, 00185-Roma, Italy. E-mail: bertola@iasi.rm.cnr.it.*
- *G. Di Battista and W. Didimo are with the Dipartimento di Informatica e Automazione, Università di Roma Tre, Via della Vasca Navale, 79, 00146 Roma, Italy. E-mail: {gdb, didimo}@dia.uniroma3.it.*

Manuscript received 22 Jan. 2000; accepted 26 Apr. 2000.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number 109025.

0018-9340/00/$10.00 © 2000 IEEE

"real-life" software engineering and database applications with the number of vertices ranging from 10 to 100. The experiments provide a detailed quantitative evaluation of the performance of four algorithms and show that they exhibit trade-offs between "aesthetic" properties (e.g., crossings, bends, edge length) and running time. The algorithm GIOTTO (topology-shape-metrics with Tamassia's algorithm in the orthogonalization step) performs better than the others in most aesthetics taken into account (e.g., number of bends, area, number of crossings, total edge length, screen ratio) at the expense of a worst time performance.

Other examples of experimental work in graph drawing follow: The performance of four planar straight-line drawing algorithms is compared in [18]. Himsolt [16] presents a comparative study of 12 graph drawings algorithms; the algorithms selected are based on various approaches (e.g., force-directed, layering, and planarization) and use a variety of graphic standards (e.g., orthogonal, straight-line, polyline). The experiments are conducted with the graph drawing system GraphEd [16]. The test suite consists of about 100 graphs. Brandenburg et al. [3] compare five "force-directed" methods for constructing straight-line drawings of general undirected graphs. Jünger and Mutzel [19] investigate crossing minization strategies for straight-line drawings of 2-layer graphs, and compare the performance of popular heuristics for this problem.

In this paper, we present the following results: Let G be a biconnected planar graph such that each vertex has degree at most 4 (*4-planar graph*).

- We describe a branch-and-bound algorithm that computes an orthogonal representation of G with the minimum number of bends in the variable embedding setting. The algorithm is based on:
  - Several new methods for computing lower bounds on the number of bends of a planar graph (Section 3). Such methods give new insights on the relationships between the structure of the triconnected components of a graph and the number of bends of an orthogonal representation.
  - A new enumeration schema that allows us to enumerate without repetitions all the planar embeddings of G (Section 4). Such an enumeration schema exploits the capability of SPQR-trees [7], [8] in implicitly representing the embeddings of a planar graph.
- We test our algorithm against a large test suite of randomly generated graphs with up to 100 vertices and compare the experimental results with the best state-of-the-art results (Section 5). Our experiments show:
  - An average improvement in the number of bends of 20 percent over the drawings constructed by the technique of [25].

    Our experiments also show several cases where the improvement in the number of bends is much more substantial. Fig. 1 shows two orthogonal drawings of the same graph. The one in Fig. 1a has the minimum number of bends within the given embedding, while the one in Fig. 1b has been computed by our algorithm and has the minimum number of bends among all of the possible embeddings.
  - An improvement of other quality measures of the drawing that are affected from the number of bends (e.g., area and edge length).
  - A required CPU-time that is affordable in most graphs of the test suite and that can be somehow "predicted" by precomputing the number of embeddings of the given graph.
- Also, our algorithm can be applied on all biconnected components of graphs. This yields a reasonable heuristic for reducing the number of bends in connected graphs.

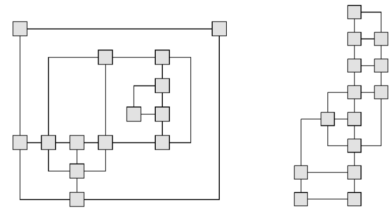

Fig. 1. (a) An orthogonal drawing with the minimum number of bends within a given embedding and (b) an orthogonal drawing of the same graph with the minimum number of bends over all the possible embeddings.

Furthermore, we study:

- how to apply our techniques for constructing drawings where constraints are specified on the shape of edges or on the angles around vertices (Section 6); and
- how to extend our approach to deal with graphs having vertices with degree greater than 4 (Section 7).

2 Preliminaries

We assume familiarity with planarity and connectivity of graphs [10], [22]. We also assume some familiarity with graph drawing [4].

A separating k-set of a graph G is a set of k vertices whose removal increases the number of connected components of G. Separating 1-sets and 2-sets are called *cutvertices* and *separation pairs*, respectively. A connected graph is said to be *biconnected* if it has no cutvertices.

Let G be a planar graph. An *embedded planar graph* G_ϕ is an equivalence class of planar drawings of G with the same ordering for the adjacency lists of vertices and with the same external face. Such a choice ϕ for an ordering of the adjacency lists and of external face is called *planar embedding* of G. Since we consider only planar graphs, we use the term *embedding* instead of *planar embedding*. A *4-planar graph* is a planar graph such that each vertex has degree at most 4.

A *planar orthogonal drawing* (or simply *orthogonal drawing*) of a planar graph maps each edge into a chain of horizontal and vertical segments. An *orthogonal grid drawing* is an orthogonal drawing such that vertices and bends along the edges have integer coordinates.

An *orthogonal representation* is an equivalence class of orthogonal drawings such that all the drawings of the class have the same sequence of left and right turns (bends) along the edges and two edges incident at a common vertex determine the same angle. Of course, all the orthogonal drawing of the same orthogonal representation have the same number of bends. Given an orthogonal representation with b bends of an n-vertex planar graph, an orthogonal grid drawing of the class can be easily constructed in $O(n+b)$ time and $O((n+b)^2)$ area by using the algorithm in [25].

An orthogonal representation of an embedded 4-planar graph G is *optimal* if it has the minimum number of bends among all possible orthogonal representations of G with the given embedding. An orthogonal representation of a 4-planar graph G is *optimal* if it has the minimum number of bends among all the possible orthogonal representations of G (considering all the possible embeddings).

A *maximal split component* of G with respect to a split pair $\{u,v\}$ (or, simpler, a maximal split component of $\{u,v\}$) is either an edge (u,v) or a maximal subgraph C of G such that C contains u and v and $\{u,v\}$ is not a split pair of C. A vertex w distinct from u and v belongs to exactly one maximal split component of $\{u,v\}$. We call the *split component* of $\{u,v\}$ a subgraph of G that is the union of any number of maximal split components of $\{u,v\}$.

Suppose $G_1,\ldots,G_k$ are some pairwise edge disjoint split components of G with respect to split pairs $\{u_1,v_1\}\ldots\{u_k,v_k\}$, respectively. The graph G' obtained by substituting each of $G_1,\ldots,G_k$ with *virtual edges* $(u_1,v_1)\ldots(u_k,v_k)$ is a *partial graph* of G. We denote by E^{virt} ($E^{nonvirt}$) the set of virtual (nonvirtual) edges of G'. We also say that G_i is the *pertinent graph* of (u_i,v_i) and that (u_i,v_i) is the *representative edge* of G_i.

Let ϕ be an embedding of G and let ϕ' be an embedding of G'. We say that ϕ *preserves* ϕ' if $G'_{\phi'}$ can be obtained from G_ϕ by substituting each component G_i with its representative edge.

In the following, we summarize SPQR-trees. For more details, see [7], [8]. An example of an SPQR-tree is shown in Fig. 2. SPQR-trees are closely related to the classical decomposition of biconnected graphs into triconnected components [17].

Let $\{s,t\}$ be a split pair of G. A *maximal split pair* $\{u,v\}$ of G with respect to $\{s,t\}$ is a split pair of G distinct from $\{s,t\}$ such that, for any other split pair $\{u',v'\}$ of G, there exists a split component of $\{u',v'\}$ containing vertices u, v, s, and t.

Let $e(s,t)$ be an edge of G, called *reference edge*. The *SPQR-tree* $\mathcal{T}$ of G with respect to e describes a recursive decomposition of G induced by its split pairs. Tree $\mathcal{T}$ is a rooted ordered tree whose nodes are of four types: S, P, Q, and R. Each node μ of $\mathcal{T}$ has an associated biconnected multigraph, called the *skeleton* of μ and denoted by $skeleton(\mu)$. Also, it is associated with an edge of the skeleton of the parent ν of μ, called the *virtual edge* of μ in $skeleton(\nu)$. Tree $\mathcal{T}$ is recursively defined as follows.

- `Trivial Case:` If G consists of exactly two parallel edges between s and t, then $\mathcal{T}$ consists of a single Q-node whose skeleton is G itself.

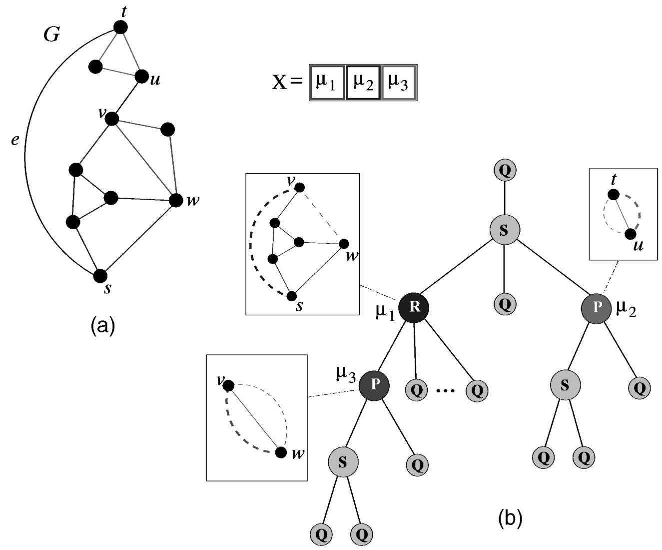

Fig. 2. (a) A planar biconnected graph G. (b) SPQR-tree $\mathcal{T}$ of G equipped with the skeletons of some nodes.

- `Parallel Case:` If the split pair $\{s,t\}$ has at least three maximal split components $G_0,\cdots,G_{k-1}$ ($k\geq 3$), the root of $\mathcal{T}$ is a P-node μ. Graph $skeleton(\mu)$ consists of k parallel edges between s and t, denoted $e_0,\cdots,e_{k-1}$, with $e_0=e$ (see Fig. 3a).
- `Series Case:` If the split pair $\{s,t\}$ has exactly two maximal split components, one of them is the reference edge e and we denote with G' the other maximal split component. If G' has cutvertices $c_1,\cdots,c_{k-1}$ ($k\geq 2$) that partition G into its blocks $G_1,\cdots,G_k$, in this order from s to t, the root of $\mathcal{T}$ is an S-node μ. Graph $skeleton(\mu)$ is the cycle $e_0,e_1,\cdots,e_k$, where $e_0=e$, $c_0=s$, $c_k=t$, and e_i connects c_{i-1} with c_i ($i=1\cdots k$) (see Fig. 3b).
- `Rigid Case:` If none of the above cases applies, let $\{s_1,t_1\},\cdots,\{s_k,t_k\}$ be the maximal split pairs of G with respect to $\{s,t\}$ ($k\geq 1$) and, for $i=1,\cdots,k$, let G_i be the union of all the maximal split components of $\{s_i,t_i\}$ but the one containing the reference edge e. The root of $\mathcal{T}$ is an R-node μ. Graph $skeleton(\mu)$ is the triconnected graph obtained from G by replacing each subgraph G_i with the edge e_i between s_i and t_i (see Fig. 3c).

Except for the trivial case, μ has children $\mu_1,\cdots,\mu_k$, in this order, such that μ_i is the root of the SPQR-tree of graph $G_i\cup e_i$ with respect to reference edge e_i ($i=1,\cdots,k$). Edge e_i is said to be the *virtual edge* of node μ_i in $skeleton(\mu)$ and of node μ in $skeleton(\mu_i)$. Graph G_i is called the *pertinent graph* of node μ_i, and of edge e_i.

The tree $\mathcal{T}$ so obtained has a Q-node associated with each edge of G, except the reference edge e. We complete the SPQR-tree by adding another Q-node, representing the reference edge e, and making it the parent of μ so that it becomes the root.

The skeletons of the nodes of $\mathcal{T}$ are homeomorphic to subgraphs of G. The SPQR-trees of G with respect to different reference edges are isomorphic and are obtained one from the other by selecting a different Q-node as the

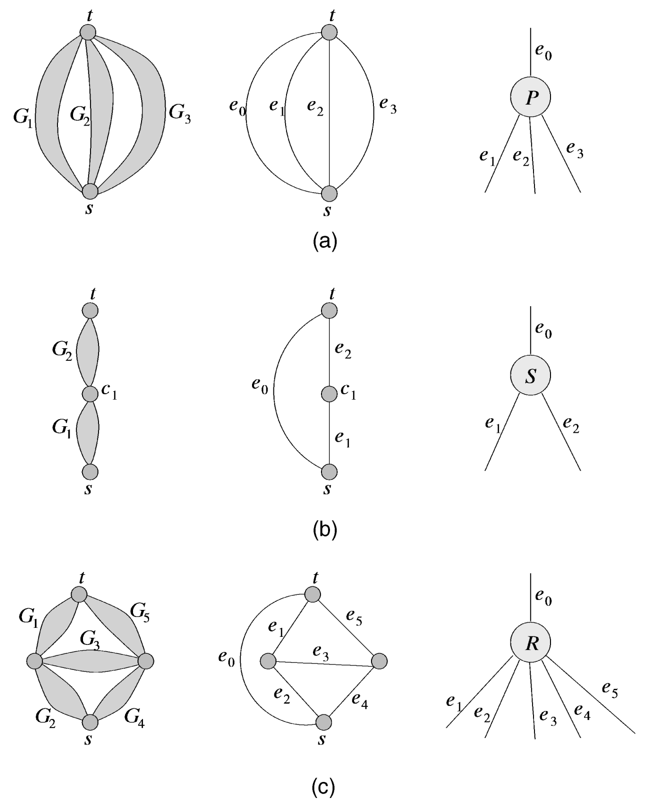

Fig. 3. (a) Parallel decomposition. (b) Series decomposition. (c) Rigid decomposition.

root. Hence, we can define the *unrooted SPQR-tree* of G without ambiguity.

The SPQR-tree $\mathcal{T}$ of a graph G with n vertices and m edges has m Q-nodes and $O(n)$ S-, P-, and R-nodes. Also, the total number of vertices of the skeletons stored at the nodes of $\mathcal{T}$ is $O(n)$.

A graph G is planar if and only if the skeletons of all the nodes of the SPQR-tree $\mathcal{T}$ of G are planar. An SPQR-tree $\mathcal{T}$ rooted at a given Q-node can be used to represent all the planar embeddings of G having the reference edge (associated with the Q-node at the root) on the external face. Namely, any embedding can be obtained by:

- Selecting one of the two possible flips of the skeleton of each R-node around its poles; and
- Selecting a permutation of the children of each P-node with respect to their common poles.

3 Lower Bounds for Orthogonal Representations

In this section, we provide new techniques for computing lower bounds on the number of bends of orthogonal representations. These techniques are illustrated by Theorem 1 and Theorem 2, which assume that a "partially" embedded graph G is given. Namely, from the hypothesis that only a subset of split components of G have a fixed embedding, Theorem 1 and Theorem 2 provide two different strategies to compute lower bounds on the number of bends of any orthogonal representation of G that preserve the "partial" embedding of G. Intuitively, denote by $G_1, \dots G_k$ the set of disjoint split components of G that do not have a fixed embedding:

- The strategy in Theorem 1 consists of replacing $G_1, \dots G_k$ with virtual edges $e_1, \dots, e_k$ of "zero-bend cost." The graph G' so obtained has a fixed planar embedding and we compute an optimal orthogonal representation H' of G' within its embedding. Bends on virtual edges are not counted. Also, if we assume that a precomputed lower bound b_i to the number of bends of any orthogonal representation of G_i $(i = 1, \dots, k)$ is known, we can sum all the b_i $(i = 1, \dots, k)$ with the number of bends of H', so obtaining a lower bound to the number of bends of any orthogonal representation of G within the given partial embedding. We use Property 1 and Lemma 1 to prove Theorem 1.
- The strategy in Theorem 2 consists of replacing each one of G_i $(i = 1, \dots, k)$ with a simple path p_i in G_i. Since each p_i has only one embedding and since the graph G' obtained after the replacement is a subgraph of G, we can easily prove that the number of bends of an optimal orthogonal representation of G' (within its embedding) is a lower bound to the number of bends of any orthogonal representation of G.

Let $G = (V, E)$ be a biconnected 4-planar graph and H be an orthogonal representation of G, we denote by $b(H)$ the total number of bends of H and by $b_{E'}(H)$ the number of bends along the edges of E' $(E' \subseteq E)$.

Property 1. *Let $G_i = (V_i, E_i)$, $i = 1, \dots, k$ be k subgraphs of G such that $E_i \cap E_j = \emptyset$, $i \neq j$, and $\cup_{i=1,\dots,k} E_i = E$. Let H_i be an optimal orthogonal representation of G_i and let H be an orthogonal representation of G. We have*

$$b(H) \geq \sum_{i=1,\dots,k} b(H_i).$$

Let G' be a partial graph of G with split components $G_1, \dots, G_k$. Let ϕ' be an embedding of G' and ϕ be an embedding of G that preserves ϕ'. See Fig. 4a, where G_1 and G_2 are two split components, and Fig. 4b, where G_1 and G_2 are substituted by virtual edges. Let $H'_{\phi'}$ be an orthogonal representation of $G'_{\phi'}$.

Suppose $H'_{\phi'}$ is such that $b_{E^{nonvirt}}(H'_{\phi'})$ is minimum and consider an optimal orthogonal representation H_ϕ of G_ϕ.

Lemma 1.

$$b_{E^{nonvirt}}(H'_{\phi'}) \leq b_{E^{nonvirt}}(H_\phi).$$

Proof. Suppose, for a contradiction, that

$$b_{E^{nonvirt}}(H'_{\phi'}) > b_{E^{nonvirt}}(H_\phi).$$

We show that this implies the existence of an orthogonal representation of $G'_{\phi'}$ with a number of bends along the nonvirtual edges that is less than the one of $H'_{\phi'}$.

For each component G_i of G, consider a simple path p_i connecting its split pair. Observe that path p_i is represented in H_ϕ by a simple polygonal line containing some vertices of G_i.

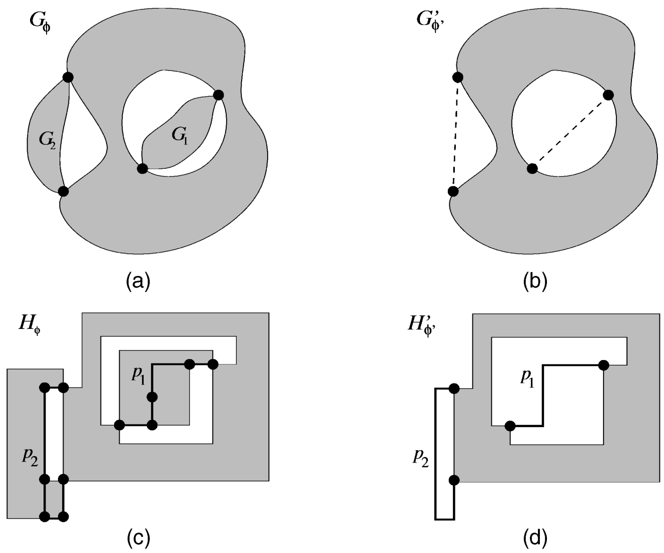

Fig. 4. Illustration of Lemma1. (a) G_1 and G_2 are two split components. (b) G_1 and G_2 are substituted by virtual edges. (c) An orthogonal representation $H_{\phi'}$ of G' with embedding ϕ'. (d) A unique polygonal line.

We can construct from H_ϕ (see Fig. 4c) an orthogonal representation $\overline{H}_{\phi'}$ of G' with embedding ϕ' as follows: First, for each G_i, we erase all the edges and vertices that do not belong to p_i. Second, we "absorb" the vertices along p_i into their incident edges in such a way to obtain a unique polygonal line. See Fig. 4d.

Since $b_{E^{nonvirt}}(\overline{H}_{\phi'}) = b_{E^{nonvirt}}(H_\phi)$, we have that $\overline{H}_{\phi'}$ has fewer bends along nonvirtual edges than $H'_{\phi'}$, a contradiction. $\square$

From Property 1 and Lemma 1, a first lower bound follows.

Theorem 1. *Let $G = (V, E)$ be a biconnected 4-planar graph and let $G'_{\phi'}$ be an embedded partial graph of G. For each virtual edge e_i of G', $i = 1, \dots, k$, let b_i be a lower bound on the number of bends of the pertinent graph G_i of e_i. Consider an orthogonal representation $H'_{\phi'}$ of $G'_{\phi'}$ such that $b_{E^{nonvirt}}(H'_{\phi'})$ is minimum. Let ϕ be an embedding of G that preserves ϕ'. Consider any orthogonal representation H_ϕ of G_ϕ. We have that:*

$$b(H_\phi) \geq b_{E^{nonvirt}}(H'_{\phi'}) + \sum_{i=1,\dots,k} b_i.$$

Remark 1. *An orthogonal representation $H'_{\phi'}$ of $G'_{\phi'}$ such that $b_{E^{nonvirt}}(H'_{\phi'})$ is minimum can be constructed by using the algorithm in [25].*

In fact, the algorithm in [25] allows computing an orthogonal representation with the minimum number of bends of an embedded 4-planar graph by mapping the problem to a minimum cost flow problem on a network $\mathcal{N}$. The nodes of $\mathcal{N}$ are the vertices and the faces of the graph. The arcs of $\mathcal{N}$ join faces and vertices. In our case, in order to determine $H'_{\phi'}$, when two adjacent faces f and g share a virtual edge, we set the costs of the (two) edges between f and g to zero.

A second lower bound is described in the following theorem.

Theorem 2. *Let G_ϕ be an embedded biconnected 4-planar graph and let $G'_{\phi'}$ be an embedded partial graph of G, such that ϕ preserves ϕ'. For each virtual edge e_i of G', $i = 1, \dots, k$, let G_i be the pertinent graph of e_i. Derive from $G'_{\phi'}$ the embedded graph $\overline{G}_{\phi'}$ by substituting each virtual edge e_i of G', $i = 1, \dots, k$, with any simple path of G_i from u_i to v_i. Consider an optimal orthogonal representation $\overline{H}_{\phi'}$ of $\overline{G}_{\phi'}$ and an orthogonal representation H_ϕ of G_ϕ. Then, we have that:*

$$b(H_\phi) \geq b(\overline{H}_{\phi'}).$$

Proof. Let $\tilde{H}_{\phi'}$ be an orthogonal representation obtained from H_ϕ of G_ϕ by removing, for each G_i, all the edges but those on the path from u_i to v_i. $\tilde{H}_{\phi'}$ is an orthogonal representation of $\overline{G}_{\phi'}$ such that $b(H_\phi) \geq b(\tilde{H}_{\phi'})$. Since $\overline{H}_{\phi'}$ is an optimal orthogonal representation of $\overline{G}_{\phi'}$, we have:

$$b(H_\phi) \geq b(\tilde{H}_{\phi'}) \geq b(\overline{H}_{\phi'}).$$

$\square$

The next corollary allows us to combine the lower bounds of Theorems 1 and 2 into a hybrid technique.

Corollary 1. *Let G_ϕ be an embedded biconnected 4-planar graph and let $G'_{\phi'}$ be an embedded partial graph of G, such that ϕ preserves ϕ'. For each virtual edge e_i of G', $i = 1, \dots, k$, let G_i be the pertinent graph of e_i. Consider a subset F^{virt} of the set of the virtual edges of G' and derive from $G'_{\phi'}$ the graph $G''_{\phi'}$ by substituting each edge $e_i \in F^{virt}$ with any simple path of G_i from u_i to v_i. Denote by E_p the set of edges of G introduced by such substitution. For each $e_j \in E^{virt} - F^{virt}$, let b_j be a lower bound on the number of bends of the pertinent graph G_j of e_j. Consider an orthogonal representation $H''_{\phi'}$ of $G''_{\phi'}$ such that $b_{E^{nonvirt} \cup E_p}(H''_{\phi'})$ is minimum. Let H_ϕ be any orthogonal representation of G_ϕ, we have that:*

$$b(H_\phi) \geq b_{E^{nonvirt} \cup E_p}(H''_{\phi'}) + \sum_{e_j \in E^{virt} - F^{virt}} b_j.$$

4 A Branch and Bound Strategy

Let G be a biconnected 4-planar graph. In this section, we describe a technique for enumerating all the possible planar embeddings of G and rules to avoid examining all of them in the computation of an orthogonal representation of G with the minimum number of bends.

4.1 Enumeration Schema

The enumeration exploits the SPQR-tree $\mathcal{T}$ of G. Namely, we enumerate all the planar embeddings of G with a certain edge e on the external face by rooting $\mathcal{T}$ at e and exploiting the capacity of $\mathcal{T}$ rooted at e in implicitly representing such embeddings. A complete enumeration is done by considering all the possible edges and rooting $\mathcal{T}$ at all of them.

We encode all the possible embeddings of G with edge e on the external face with an r-tuple X of variables in the following way:

- We consider the SPQR-tree $\mathcal{T}$ of G and root $\mathcal{T}$ at e. We visit $\mathcal{T}$ such that a node is visited after its parent, e.g., breadth first or depth first. The visit induces an ordering $\mu_1, \ldots, \mu_r$ of the P- and R-nodes of $\mathcal{T}$.
- We define an r-tuple of variables $X = (x_1, \ldots, x_r)$ that are in one-to-one correspondence with the P- and R-nodes $\mu_1, \ldots, \mu_r$ of $\mathcal{T}$. Each x_i can assume either values in the set of the nonnegative integer numbers or value "$-$" (null value).
  - Each variable x_i of X, corresponding to an R-node μ_i, can be set either to 0 or to 1 (corresponding to the two possible embeddings of $skeleton(\mu_i)$) or to null. The null value in variable x_i means that the embedding of $skeleton(\mu_i)$ is not specified.
  - Each variable x_j of X, corresponding to a P-node μ_j, can be set either to a value in the range $0, \ldots, (\deg(\mu_j)! - 1)$ or to the null value, where $\deg(\mu_j)$ is the number of children of μ_j. Observe that, in a 4-planar graph, we have $2 \leq \deg(\mu_j) \leq 3$. Each value of x_j corresponds to one of the possible permutations of the edges of $skeleton(\mu_j)$ around its two vertices. Again, the null value means that the embedding of the skeleton is not specified.

Fig. 2a shows a biconnected planar graph and Fig. 2b shows its SPQR-tree $\mathcal{T}$. The skeletons of the nodes are also shown. The R- and P-nodes of $\mathcal{T}$ are highlighted. In this case, X is a 3-ple where x_1, x_2, and x_3 are in correspondence with μ_1, μ_2, and μ_3, respectively.

4.2 Search Tree

A *search tree* $\mathcal{B}$ for our problem is defined as follows.

- Each node β of $\mathcal{B}$ corresponds to a different setting X_β of $X = (x_1, \ldots, x_r)$. Such a setting is partitioned into two (one of them possibly empty) subsequences $x_1, \ldots, x_h$ and $x_{h+1}, \ldots, x_r$. The elements of the first subsequence have integer values (do not have null values), while the elements in the second subsequence have null values only.
- The leaves of $\mathcal{B}$ are in correspondence with settings of X without null values.
- Internal nodes of $\mathcal{B}$ are in correspondence with settings of X with at least one null value.
- The setting of the root of $\mathcal{B}$ consists of null values only.
- The nonleaf node β with $X_\beta = (\overline{x}_1, \ldots, \overline{x}_h, -, \ldots, -)$ has one child for each possible integer value of x_{h+1}. The child β' of β corresponding to value $\tilde{x}_{h+1}$ has $X_{\beta'} = (\overline{x}_1, \ldots, \overline{x}_h, \tilde{x}_{h+1}, -, \ldots, -)$.

Property 2. *The leaves of $\mathcal{B}$ are in one-to-one correspondence with the planar embeddings of G with edge e on the external face.*

Fig. 5 shows the search tree $\mathcal{B}$ associated with the SPQR-tree of Fig. 2b. The leaves (whose 3-ple do not have null values) are equipped with the corresponding planar embeddings.

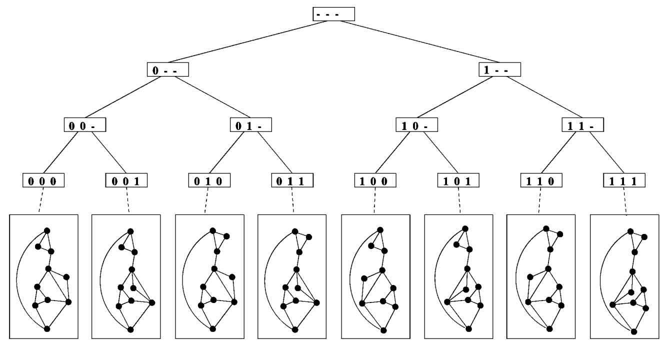

Fig. 5. The search tree of the SPQR-tree in Fig. 2b. The leaves are equipped with the corresponding planar embeddings.

Since each internal node of $\mathcal{B}$ has at least two children, by Property 2, we have that the internal nodes of $\mathcal{B}$ are less than the number of embeddings of G with edge e on the external face.

Lemma 2. *The nodes of $\mathcal{B}$ are in one-to-one correspondence with the embedded partial graphs of G with edge e on the external face.*

Proof. Given a node of $\mathcal{B}$, we first show the corresponding embedded partial graph. The embedded partial graph G_β of G associated with node β of $\mathcal{B}$ with $X_\beta = (\overline{x}_1, \ldots, \overline{x}_h, -, \ldots, -)$ is obtained as follows: First, set G_β to $skeleton(\mu_1)$ embedded according to x_1. Second, substitute each virtual edge e_i $(2 \leq i \leq h)$ of μ_1 with the skeleton of the child μ_i of μ_1, embedded according to $\overline{x}_i$. Then, recursively substitute virtual edges with embedded skeletons until all the skeletons in $\{skeleton(\mu_1), \ldots, skeleton(\mu_h)\}$ have been used.

The fact that the embedded partial graphs associated with two distinct nodes β_1 and β_2 of $\mathcal{B}$ are distinct follows immediately from the presence of at least one different value in X_{β_1} and X_{β_2}.

The fact that, given an embedded partial graph G' of G with edge e on the external face, it is possible to find a node of $\mathcal{B}$ associated with G' follows immediately from the definition of $\mathcal{B}$. □

Fig. 6 shows the same search tree of Fig. 5. The internal nodes are equipped with the corresponding embedded partial graphs.

Our computation works as follows: We visit $\mathcal{B}$ breadth-first starting from the root. At each internal node β of $\mathcal{B}$ with setting X_β, we compute a lower bound and an upper bound of the number of bends of any orthogonal representation of G with the embedding (partially) specified by X_β. The current optimal solution is updated accordingly. The children of β are not visited if the lower bound is greater than the current optimum.

For each β, lower bounds and upper bounds are computed as follows, by using the results presented in Section 3.

1. We construct the embedded partial graph G_β corresponding to β (see Lemma 2).
2. We compute lower bounds.

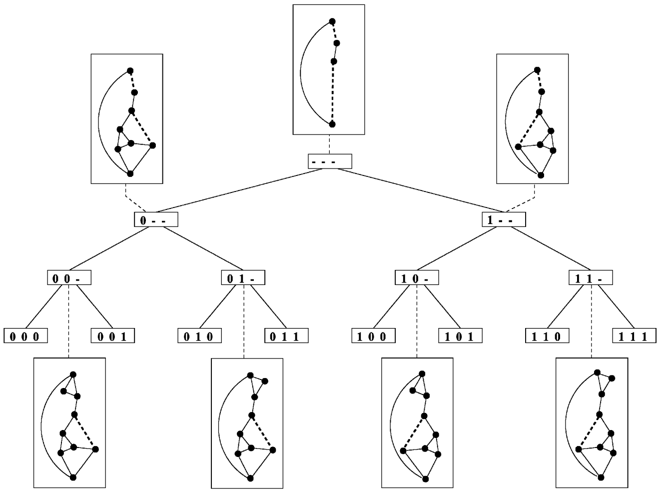

Fig. 6. The search tree of the SPQR-tree in Fig. 2b. The internal nodes are equipped with the corresponding embedded partial graphs. Virtual edges are dotted.

- Let E^{virt} be the set of virtual edges of G_β. For each $e_i \in E^{virt}$, consider the pertinent graph G_i and a lower bound b_i on the number of bends of G_i. See Theorem 1. Denote by F^{virt} the set of edges e_i such that $b_i = 0$. We replace in G_β each edge $e_i \in F^{virt}$ with any simple path of G_i from u_i to v_i, so deriving a new embedded graph G'_β. Denote by E_p the set of edges of G introduced by the substitution (observe that E_p may be empty).
- By Remark 1, we can apply Tamassia's algorithm on G'_β, assigning zero costs to the edges of the dual graph associated to the virtual edges. We obtain an orthogonal representation H'_β of G'_β with minimum number of bends on the set $E^{nonvirt} \cup E_p$. Let $b_{E^{nonvirt} \cup E_p}(H'_\beta)$ be such a number of bends. Then, by using Corollary 1, we compute the lower bound L_β at node β, as

$$L_\beta = b_{E^{nonvirt} \cup E_p}(H'_\beta) + \sum_{e_i \in E^{virt} - F^{virt}} b_i.$$

  Lower bounds b_i can be precomputed with a suitable preprocessing strategy that will be illustrated afterward.

3. We compute upper bounds. Namely, we consider the embedded partial graph G_β and complete it to a pertinent embedded graph G_ϕ. The embedding of G_ϕ is obtained by substituting the null values of X_β with integer values choosen in a random way over the possible values. Then, we apply Tamassia's algorithm to G_ϕ, so obtaining the upper bound. We also avoid multiple generations of the same embedded graph in completing the partial graph.

The precomputation of the b_i lower bounds is done in two phases:

1. For each P- and R-node μ_i, we apply Tamassia's algorithm several times to compute an optimal orthogonal representation of $skeleton(\mu_i)$. The algorithm is applied for all the possible embeddings of $skeleton(\mu_i)$ with the reference edge on the external face. Also, the bends on the virtual edges are considered as zero cost bends. We choose the orthogonal representation with the minimum number of bends. Let $\overline{b}_i$ be such a number. We label μ_i with $\overline{b}_i$. For each Q- and S-node μ_i, $\overline{b}_i = 0$.
2. We visit $\mathcal{T}$ bottom-up. At each node μ_i we compute b_i as the sum of $\overline{b}_i$ plus the values of the b_j for each child μ_j of μ_i.

Of course, to compute the optimal solution, we must apply the above algorithm over all the possible choices of the reference edge e. For each choice, we root $\mathcal{T}$ at a different Q-node. Observe that this strategy can lead to considering the same embedding several times. To avoid this, if an edge e that has already been reference edge appears on the external face of some G_β under computation, since we have already explored all the embeddings with e on the external face, we cut all the descendants of β from the search tree. In this way, each embedding is considered exactly once.

5 Experimental Setting and Computational Results

The branch and bound algorithm presented above has been implemented with the C++ language using the Leda [21] platform and it has been inserted into the GDToolkit system (www.dia.uniroma3.it/~gdt). Further, to support experiments, we have developed a new graphical interface that shows an animation of the algorithm in which the SPQR-tree is rooted at different reference edges. Such an interface displays the relationship between the SPQR-tree and the search tree and the evolution of the search tree. It allows us to visualize the skeletons of the nodes. Of course, to show the animation, the graphical interface exploits several graph drawing algorithms.

5.1 The Test Suite

We have tested our algorithm against a randomly generated test suite (available on the Web at www.dia.uniroma3.it/~didimo) overall consisting of 500 (multi)graphs (multiple-edges allowed). The test suite has been generated as follows: It is well-known that any planar biconnected graph can be generated from the triangle graph by means of a sequence of *Insert-Vertex* and *Insert-Edge* operations. Insert-Vertex subdivides an existing edge into two new edges separated by a new vertex. Insert-Edge inserts a new edge between two existing vertices that can be placed on the same face. Hence, we have implemented a generation mechanism that works as follows.

- A graph is generated with a sequence of steps. The starting graph is a triangle.
- At each step, either an Insert-Edge or an Insert-Vertex operation is performed. An Insert-Edge operation is performed with probability p and an Insert-Vertex is performed with probability $1 - p$. Actually, the probabilities of really performing the two operations are slightly different, as will be clear below.
- If we perform an Insert-Vertex, then we randomly select one already existing edge and split it.

TABLE 1
Average Density by Number of Vertices and by Insert-Edge Operation Probability

	10	20	30	40	50	60	70	80	90	100
0.1	1.05	1.08	1.09	1.09	1.12	1.12	1.10	1.13	1.11	1.12
0.2	1.23	1.24	1.25	1.19	1.22	1.21	1.22	1.24	1.24	1.23
0.3	1.31	1.43	1.48	1.39	1.41	1.43	1.43	1.41	1.41	1.49
0.4	1.47	1.61	1.53	1.68	1.58	1.60	1.62	1.60	1.61	1.62
0.5	1.55	1.74	1.75	1.73	1.76	1.81	1.84	1.84	1.84	1.82

TABLE 2
Average Number of Embeddings by Number of Vertices and by Insert-Edge Operation Probability

	10	20	30	40	50	60	70	80	90	100
0.1	2.3	6.9	14.4	14.8	31.8	41.2	37.6	97	108.9	69.8
0.2	6.8	29.3	54.7	82.4	82.6	102.6	151.3	257.5	294.2	480.2
0.3	27.7	90.2	265.6	194.8	607.5	1524	1993.2	4212.4	3709.6	49464.8
0.4	31.6	344.2	268	8492.8	2918.4	89886.4	81362.4	19102.4	62483.2	375273.6
0.5	52.5	336.6	4710	5537.6	15536	5411867.2	11829350.4	246648729.6	131872768	101412044.8

- If we perform an Insert-Edge, then we randomly select one pair of already existing vertices (say u and v) and apply the following checks.
  - If $\deg(u) < 4$ and $\deg(v) < 4$ and the graph remains planar after the insertion of edge (u, v), then we add (u, v).
  - Else, we randomize a new pair of vertices and retry.

  After unsuccessfully trying with all the possible pairs of vertices we perform an Insert-Vertex instead of an Insert-Edge.
- The generation stops when a given number n of vertices has been reached.

We have generated the entire test suite by generating 10 graphs for each possible value of the pair p, n, where p can assume values in the set $\{0.1, 0.2, 0.3, 0.4, 0.5\}$ and n can assume values in the set $\{10, 20, \dots, 100\}$. Table 1 shows the density of the generated graphs. The average density d is presented for the graphs with the same values of p and n. Observe that, for 4-planar graphs, $1 \leq d \leq 2$ and that the density of our graphs grows linearly with p. We roughly have $d = 1.1$ for $p = 0.1$, $d = 1.2$ for $p = 0.2$, $d = 1.4$ for $p = 0.3$, $d = 1.6$ for $p = 0.4$, and $d = 1.8$ for $p = 0.5$. In the following tables, we shall present the experimental results by classifying graphs by n and p. Because of the relationship between d and p, such tables can be read as they were by n and d.

Table 2 and Fig. 7 show the number of planar embeddings of the generated graphs. Such a number is somehow a measure of the difficulty of the problem. For reducing the total computation time of our experiments, we have tested our algorithm only over the graphs of the test suite that have at most $200,000$ embeddings. In this way, we did not run the algorithm on 41 graphs. Table 3 shows the number of graphs that have been tested by number of vertices and by Insert-Edge operation probability. Observe that all the 41 discarded graphs have high density.

5.2 The Experimental Setting

For each drawing constructed by the algorithm, we have measured the following quality parameters: total number of edge bends, maximum number of bends on an edge, standard deviation of the number of edge bends, area of the smallest rectangle with horizontal and vertical sides covering the drawing, total edge length, maximum length of an edge, standard deviation of the edge length. Measured performance parameters are: CPU time, total number of search tree nodes, number of visited search tree nodes. The test has been performed under the Linux operating system on a 266 MHz PentiumII personal computer with 128 MB Ram.

For the compaction phase we have used the min-cost-flow algorithm called *Tidy-Rectangle-Compact* illustrated in [4].

5.3 Results

The results of our experiments are summarized in the following tables. The tables give the percentage of improvement that we have obtained of the most important quality measures with respect to the results of the fixed embedding algorithm in [25]. The embedding that we use for the algorithm in [25] is randomly chosen.

Table 4 and Fig. 8 show an average decreasing of the total number of bends that is about 20 percent. The improvement is greater for graphs with density between 1.2 and 1.4. The percentage of improvement is computed as $100 \times ((b_{fix} - b_{opt}) / \max\{1, b_{fix}\})$, where b_{opt} and b_{fix} are the

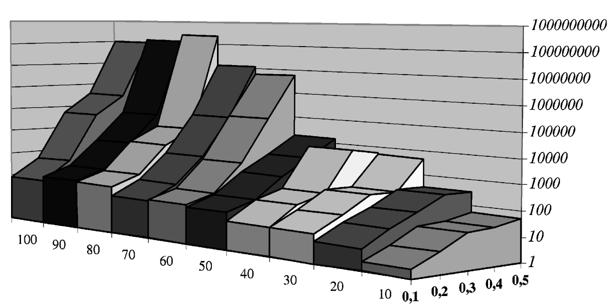

Fig. 7. Average number of embeddings by number of vertices and by Insert-Edge operation probability (log. scale).

TABLE 3
Number of Tested Graphs by Number of Vertices and by Insert-Edge Operation Probability

	10	20	30	40	50	60	70	80	90	100
0.1	10	10	10	10	10	10	10	10	10	10
0.2	10	10	10	10	10	10	10	10	10	10
0.3	10	10	10	10	10	10	10	10	10	9
0.4	10	10	10	10	10	9	9	10	9	7
0.5	10	10	10	10	10	7	4	3	2	0

number of bends obtained with our algorithm and with the algorithm in [25], respectively. Observe that if $b_{fix} = 0$, then $b_{opt} = 0$ also and, hence, the improvement is 0 percent and that if $b_{opt} = 0$ and $b_{fix} \neq 0$, then the improvement is 100 percent.

Table 5 and Fig. 9 show the best improvement obtained for each value of n and p. Observe that this number dramatically increases up to 100 percent for low densities.

The results of Table 4 and Table 5 show that, on the average, selecting a random embedding and running the algorithm in [25] is a reasonable heuristic. On the other side, there are cases where it obtains results that are very far from the optimum.

The following tables show how minimizing the number of bends may have positive effects on other quality measures. Table 6 shows the average improvement of the maximum number of bends on an edge and Table 7 shows the improvement in the best case. Observe that, although the overall improvement is good, there are some cases where it is negative. As is shown in Table 7, the improvement can be very high in some cases.

Table 8 and Table 9 show the improvement on the length of the longest edge. While in the average analysis we also have negative results, as far as the best cases are concerned we can obtain up to 80 percent of decreasing.

Concerning performance, Table 10 shows the CPU-time spent by our algorithm. Observe that, for graphs with high density and high number of vertices, the algorithm can take about three hours of CPU time.

We recall that the results presented so far concern only graphs up to 200,000 planar embeddings. To have an idea of the feasibility of the approach with graphs with a very large number of embeddings, we have tested one of the 41 discarded graphs having about 1.2 million embeddings and 100 vertices. It required about 12 hours of CPU time.

This shows that the approach is still feasible for graphs up to some millions of embeddings. However, of the 41 discarded graphs, we had 17 graphs with more than 10 million embeddings; nine of them have more than 100 million, and three of them have more than 1,000 million. Of course, this is not appropriate for an interactive system, where the user expects a quick response.

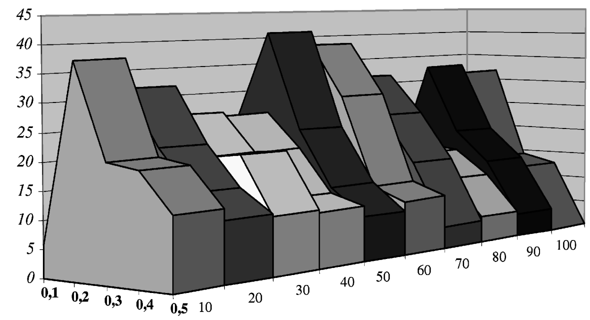

Fig. 8. Percentage decrease (average) of the number of bends by number of vertices and by Insert-Edge operation probability.

Table 11 shows the percentage of the number of search tree nodes that are visited during the branch and bound computation over the total number of search tree nodes. Observe that this number decreases when the number of vertices increases.

6 Constraints and Embeddings

Our technique allows us to easily take into account the constraints that are commonly used [4] for customizing orthogonal representations. Namely, the following constraints can be considered.

- A prescribed shape can be specified on an edge. For example, we can say that an edge (u, v) should have no bends or that the bends should only be "to the left" while following (u, v) from u to v. Further, we can say that (u, v) can have an unlimited number of bends.
- A prescribed angle can be specified to the left (or to the right) of an edge. For example, we can say that edge (u, v) should have a 180 degree angle on its left while following (u, v) from u to v.

Such constraints are naturally introduced in the branch and bound as constraints on the network flows that are constructed both for computing lower bounds and for computing upper bounds.

Observe that having the possibility of specifying constraints over our algorithm dramatically enlarges the degrees of freedom of the user in customizing a drawing. In fact, there are sets of constraints that are not satisfiable for a given embedding and that may be satisfiable for another one. Since our algorithm explores all the planar embeddings, one

TABLE 4
Percentage Decrease (Average) of the Number of Bends by Number of Vertices and by Insert-Edge Operation Probability

	10	20	30	40	50	60	70	80	90	100
0.1	6.00	10.00	10.00	25.00	25.00	10.00	10.00	4.00	13.33	0.00
0.2	37.50	32.38	27.00	26.33	41.30	38.67	32.10	17.32	33.28	31.76
0.3	20.82	22.31	19.99	19.92	23.35	28.99	24.88	16.59	20.36	14.20
0.4	19.75	15.05	20.76	12.16	13.14	12.40	15.92	11.87	14.61	12.65
0.5	13.04	11.05	10.46	10.08	8.15	9.94	4.07	4.77	4.21	

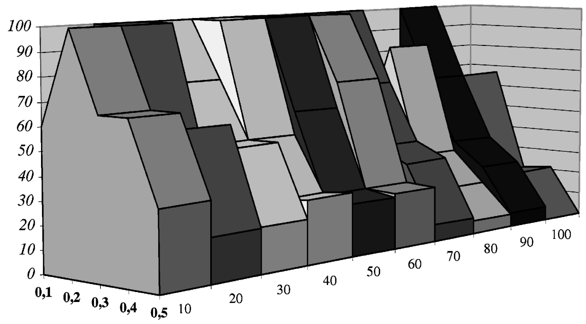

Fig. 9. Percentage decrease (best) of the number of bends by number of vertices and by Insert-Edge operation probability.

solution is found if any exists. For example, consider the graph in Fig. 10a and suppose we want to have no bends on cycle $(0,4),(4,3),(3,2),(2,1),(1,0)$. Such constraints are not satisfiable if the embedding remains the same. Fig. 10b shows an orthogonal drawing, computed by our algorithm, where the constraints are satisfied.

7 Extending the Technique for Drawing Graphs with High Degree Vertices

The techniques proposed so far have several possible extensions. In this section, we show a possible extension to deal with graphs with high degree vertices.

Two main drawing conventions have been defined in the literature to construct orthogonal drawings of embedded planar graphs with vertices with degree greater than 4. The first one consists of expanding vertices with high degree into boxes that are large enough to allow several edges to coincide on each side of the box. The size of the box increases with the degree of the vertex. See, e.g., [26]. The second one consists of representing vertices as fixed size boxes and allowing the segments representing edges to stay "very close together" if they are coincident on the same side of the box. See, e.g., [12].

7.1 Expanded Vertices

There are several ways to "expand" a vertex v to allow more than four edges to coincide on v. In [26], v is expanded to a set of predetermined structures, depending on $\deg(v)$. For example, if $\deg(v)=8$, with incident edges $e_1,\dots,e_8$ (circularly ordered according to a given embedding), v is expanded into a cycle of four vertices $v_1,\dots,v_4$ such that e_1 and e_2 coincide on v_1, e_3 and e_4 coincide on v_2, etc. Then, the algorithm for computing an orthogonal representation with the minimum number of bends is invoked with the constraint that the edges of the cycle substituting v have zero bends. In this way, the structure replacing v is drawn as a box. This strategy has two main drawbacks.

1. In the computation of the final drawing, the compaction algorithms that are used in practice [4] may "stretch" v to a very large box.
2. A degree of freedom in the replacement is the choice of "what edges coincide on what vertices." For instance, referring to the above example, if in replacing v we make incident on v_1 edges e_2 and e_3, incident on v_2 edges e_4 and e_5, etc., then we may obtain a different orthogonal representation, with a different number of bends.

The second drawback can be overcome with a slightly different replacement strategy that is commonly adopted in graph drawing systems and that is described, e.g., in [9]. In this case, if $\deg(v)>4$, then v is replaced by a cycle with $\deg(v)$ vertices and each edge incident on v is assigned to a different vertex of the cycle. This has the advantage of imposing fewer constraints on the shape of the drawing.

7.2 Nonexpanded Vertices

The possibility of using nonexpanded vertices has been extensively studied by Fößmeier and Kaufmann in [12]. Roughly speaking, the proposed model is the following:

1. Vertices are still points of the grid, but it is easier to think of them in terms of squares of half unit sides centered at grid points.
2. Two segments that are coincident on the same vertex may overlap. Observe that the angle between such segments has zero degree.
3. All the polygons representing the faces have area strictly greater than zero.
4. If two segments overlap, they are presented to the user as two very near segments.

Because of Conditions 2 and 3, each zero degrees angle is in correspondence to exactly one bend [12].

Within this model, the algorithm in [25] is not applicable. Hence, Fößmeier and Kaufmann propose a new minimum cost flow technique based on augmenting the flow network in [25] with arcs from faces to vertices. The flow on such arcs represents zero degrees angles. The resulting network is quite complicated. Also, the minimum cost flow problem associated to the network requires knowing in advance an upper bound on the number of bends. Such bound is used to set the cost of some of the arcs of the network to avoid the computation of unfeasible solutions. Further, it is unclear how to construct the final drawing starting from the orthogonal representation.

TABLE 5
Percentage Decrease (Best) of the Number of Bends by Number of Vertices and by Insert-Edge Operation Probability

	10	20	30	40	50	60	70	80	90	100
0.1	60.00	100.00	100.00	100.00	100.00	100.00	100.00	40.00	100.00	0.00
0.2	100.00	100.00	75.00	100.00	100.00	100.00	66.67	83.33	66.67	66.67
0.3	66.67	50.00	50.00	50.00	60.00	71.43	40.00	33.33	37.50	20.00
0.4	66.67	60.00	50.00	26.09	25.00	25.00	33.33	19.44	26.47	21.21
0.5	33.33	19.05	19.35	27.59	22.73	24.24	7.25	5.77	6.25	

TABLE 6
Percentage Decrease (Average) of the Maximum Number of Bends on an Edge by Number of Vertices and by Insert-Edge Operation Probability

	10	20	30	40	50	60	70	80	90	100
0.1	6.67	15.00	0.00	20.00	25.00	10.00	10.00	-10.00	10.00	0.00
0.2	40.83	21.67	8.33	20.00	31.67	33.00	11.67	1.67	18.33	6.67
0.3	9.17	8.33	20.83	8.33	16.67	22.50	21.67	1.67	18.33	12.96
0.4	1.67	-5.83	26.83	2.50	9.17	11.11	-3.70	14.17	8.33	33.10
0.5	12.50	2.50	-6.33	-5.83	9.17	9.52	8.33	-8.33	16.67	

TABLE 7
Percentage Decrease (Best) of the Maximum Number of Bends on an Edge by Number of Vertices and by Insert-Edge Operation Probability

	10	20	30	40	50	60	70	80	90	100
0.1	66.67	100.00	100.00	100.00	100.00	100.00	100.00	0.00	100.00	0.00
0.2	100.00	100.00	50.00	100.00	100.00	100.00	66.67	66.67	66.67	66.67
0.3	66.67	66.67	75.00	50.00	75.00	75.00	50.00	50.00	66.67	50.00
0.4	50.00	33.33	60.00	50.00	50.00	50.00	33.33	50.00	33.33	50.00
0.5	50.00	50.00	33.33	33.33	33.33	50.50	50.50	25.00	33.33	

TABLE 8
Percentage Decrease (Average) of the Length of the Longest Edge by Number of Vertices and by Insert-Edge Operation Probability

	10	20	30	40	50	60	70	80	90	100
0.1	-7.31	-22.57	3.53	-5.70	0.21	-6.92	-13.86	-3.60	-11.29	-21.91
0.2	26.98	26.95	8.96	-6.68	20.41	21.76	8.38	11.80	12.60	-12.91
0.3	28.13	12.46	30.10	0.22	1.15	27.50	31.27	7.62	21.38	20.30
0.4	22.92	11.27	28.15	-0.41	11.96	13.00	22.76	17.86	4.92	30.07
0.5	28.47	16.65	-0.32	15.67	14.31	19.54	8.61	3.72	-95.20	

TABLE 9
Percentage Decrease (Best) of the Length of the Longest Edge by Number of Vertices and by Insert-Edge Operation Probability

	10	20	30	40	50	60	70	80	90	100
0.1	76.92	41.67	40.00	29.41	43.75	34.48	4.17	33.33	25.53	38.89
0.2	75.00	77.27	50.00	57.14	68.63	57.45	61.40	44.44	63.64	54.10
0.3	75.00	81.48	72.22	63.33	61.90	76.00	72.73	60.87	45.45	51.79
0.4	77.78	68.75	84.21	53.57	57.89	76.00	72.73	57.14	68.42	71.00
0.5	64.29	69.70	52.00	48.89	54.39	62.22	61.02	44.26	15.15	

TABLE 10
Seconds of CPU-Time (Average) by Number of Vertices and by Insert-Edge Operation Probability

	10	20	30	40	50	60	70	80	90	100
0.1	0.55	1.19	2.88	4.78	10.07	11.87	15.31	23.31	82.24	69.68
0.2	0.97	2.92	8.07	10.48	21.04	23.00	52.61	114.93	115.87	148.86
0.3	1.34	6.22	17.14	23.56	49.56	94.08	276.19	195.17	203.11	656.56
0.4	1.82	10.29	21.82	464.27	90.12	138.23	321.74	665.80	969.53	910.99
0.5	2.60	17.80	69.66	265.17	791.48	2205.48	3532.21	11834.54	4540.95	

TABLE 11
Percentage of Branch and Bound Nodes Visited by the Algorithm (Average) by Number of Vertices and by Insert-Edge Operation Probability

	10	20	30	40	50	60	70	80	90	100
0.1	4.71	0.14	0.25	0.65	7.49	0.12	0.05	0.05	0.57	0.63
0.2	11.76	12.77	8.90	5.45	11.50	0.75	0.88	2.31	1.40	1.05
0.3	11.71	8.09	5.38	2.33	9.85	1.82	2.76	2.83	2.03	1.01
0.4	15.22	6.53	5.75	2.26	1.83	2.31	1.40	1.53	0.76	0.40
0.5	23.86	12.33	3.64	2.81	2.40	2.20	2.60	1.89	0.94	

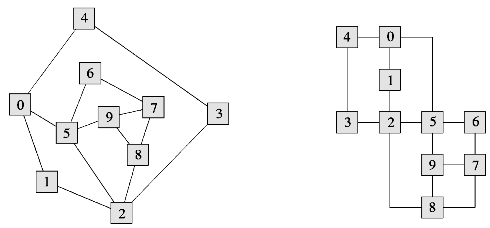

Fig. 10. (a) A planar graph. A constraint "no bends" is specified on the edges of cycle $(0,4),(4,3),(3,2),(2,1),(1,0)$. (b) An orthogonal drawing that satisfies the constraints and that has a different embedding with respect to the one in (a).

7.3 A Simple Model

We have devised a new drawing convention for orthogonal drawings of planar graphs with vertices with degree greater than four. Such a convention is very similar to the one in [12], but requires usage of a much simpler flow network and avoids the aesthetic drawbacks of the expanded vertices.

In our model, vertices and edges can be represented with the rules of [12], with two additional constraints:

- In order to distribute the edges around a vertex more uniformly, each vertex with degree greater than four has at least one incident edge on each side.
- Consider a vertex u with $\deg(u) > 4$. Consider two edges (u,v) and (u,w) incident on u and such that (u,w) follows (u,v) in the circular clockwise ordering given by the embedding. If there is a zero degree angle at u between (u,v) and (u,w), then 1) (u,w) contains at least one bend and 2) the first bend of (u,w) encountered while following (u,w) from u to w causes a right turn.

Since the drawing convention adopted in [12] is called *podevsnef* (planar orthogonal drawing with equal vertex size and nonempty faces), we call an orthogonal drawing that follows our drawing convention *simple-podevsnef*. The simple-podevsnef drawing convention has several advantages: Vertices are not expanded, the computation of the orthogonal representation can be done with a standard minimum cost flow, and the final orthogonal drawing can be computed with a standard compaction technique. An orthogonal drawing in the simple-podevsnef model is shown in Fig. 11.

The following property is a direct consequence of the definition:

Property 3. *A simple-podevsnef drawing of a planar embedded graph has at least*

$$\sum_{v:\deg(v)>4} (\deg(v) - 4)$$

bends.

In the same way that we have defined the concept of orthogonal representation as an equivalence class of orthogonal drawings, we define *simple-podevsnef representation* as an equivalence class of simple-podevsnef drawings.

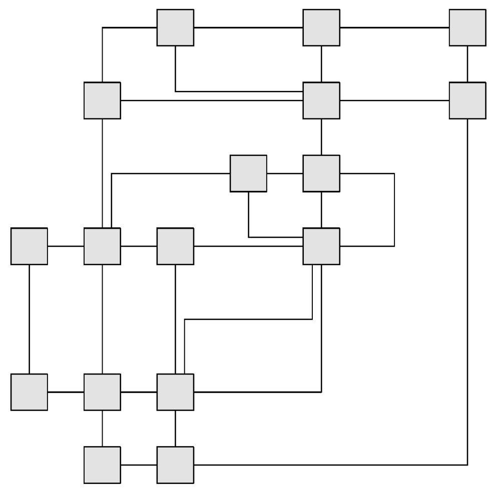

Fig. 11. An orthogonal drawing in the simple-podevsnef model.

We associate with an embedded planar graph G a flow network $\mathcal{N}$. From the minimum cost flow computed on $\mathcal{N}$, it is easy to compute a simple-podevsnef representation of G with a number of bends equal to the cost of the flow. Network $\mathcal{N}$ is as follows:

- The nodes of $\mathcal{N}$ are the vertices and the faces of G.
- For each pair of adjacent faces f_1 and f_2, we have in $\mathcal{N}$ arcs (f_1, f_2) and (f_2, f_1), both with infinite capacity and cost equal to one.
- Let u be a vertex. Consider edge (u,v) and let f_r (f_l) be the face on the right (left) of (u,v) when going from u to v. For each u and for each f_r, we have in $\mathcal{N}$:
  - If $\deg(u) \leq 4$, an arc (u, f_r) with capacity equal to $4 - \deg(u)$ and cost equal to zero;
  - If $\deg(u) > 4$, an arc (f_r, u) with capacity equal to one and cost equal to one.
- Each node of $\mathcal{N}$ corresponding to a vertex u of G supplies an amount of flow equal to $4 - \deg(u)$. Observe that, when $\deg(u) > 4$, the supply becomes a demand.
- Each node of $\mathcal{N}$ corresponding to an internal face f of G supplies an amount of flow equal to $4 - \deg(f)$, where $\deg(f)$ is equal to the number of edges traversed when doing a complete "tour" around f. Observe that, when $\deg(f) > 4$, the supply becomes a demand.
- The node of $\mathcal{N}$ corresponding to the external face h of G demands an amount of flow equal to $4 + \deg(h)$.

Lemma 3. *For each flow computed on $\mathcal{N}$, there exists a corresponding simple-podevsnef representation having a number of bends equal to the cost of the flow. For each simple-podevsnef representation, there exists a flow on $\mathcal{N}$ whose cost is equal to the number of bends of the drawing.*

Proof. We give a constructive proof similar to the one in [25].

Given a flow computed on $\mathcal{N}$, we construct a simple-podevsnef representation. We use the same notation used in the definition.

- A flow of value k, with $k = 0, \ldots, 3$, on arc (u, f_r), with $\deg(u) \leq 4$ causes an angle of $(k+1)90$ degrees at vertex u on face f_r and to the right of edge (u, v).
- A flow of value 1 on arc (f_r, u), with $\deg(u) > 4$ causes an angle of zero degrees at vertex u on face f_l and to the left of edge (u, v). Also, one bend is placed on edge (u, v). Such a bend is the first bend that is found while traversing (u, v) from u to v and causes a right turn.
- A flow of value k on arc (f_1, f_2) causes k bends on an edge e shared by f_1 and f_2. While following e leaving f_1 to the right, such angles are right turns. If f_1 and f_2 coincide, then the ambiguity can be solved arbitrarily.

The fact that the obtained representation is a simple-podevsnef representation is proven in the same way as in [25].

At this point we show how, given a simple-podevsnef representation, it is possible to construct a feasible flow of $\mathcal{N}$.

- For each angle of $90 \leq h \leq 360$ degrees between edges (u, v) and (u, w) at u, we set the value of the flow on arc (u, f_r) to $h/90 - 1$.
- For each angle of zero degrees between edges (u, v) and (u, w) at u, we set the value of the flow on arc (f_r, u) to 1.
- For each edge (u, v) between faces f_1 and f_2, follow (u, v) leaving f_1 to the right and consider the bends on (u, v) that are not associated with a zero angle at u or at v. Let h_1 be the number of the right turns and let h_2 be the number of the left turns. We set the values of the flows on arcs (f_1, f_2) and (f_2, f_1) to h_1 and h_2, respectively.

It is easy to see that the constructed flow is feasible and that its cost is equal to the number of bends of the given simple-podevsnef representation. □

Lemma 3 and [15] yield the following theorem.

Theorem 3. *Let G be an n-vertex planar embedded graph. A simple-podevsnef representation of G with the minimum number of bends can be computed in $O(n^{7/4}\log(n))$.*

Given a simple-podevsnef representation, a simple-podevsnef drawing can be constructed by exploiting the following theorem.

Theorem 4. *Given a simple-podevsnef representation H with b bends of an n-vertex planar graph, a simple-podevsnef grid drawing of H can be constructed in $O(n+b)$ time and $O((n+b)^2)$ area.*

Proof. Consider any pair $(u, v), (u, w)$ of edges that are consecutive in the adjacency list of u and that form a zero degree angle at u. Consider the two segments s_1 and s_2 of (u, v) and (u, w) (respectively) that end at u. From the definition of simple-podevsnef representation, a zero degree angle causes a bend to appear on one of the two edges that form such an angle. Therefore, one of the two segments s_1 and s_2 (say s_2) must be shorter than the other

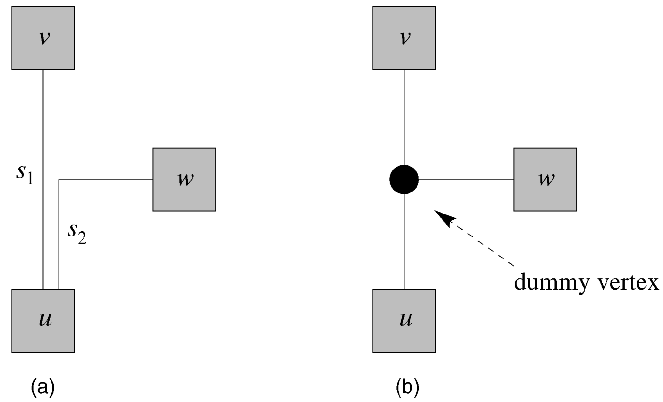

Fig. 12. Illustration of the proof of Theorem 4. (a) The edges (u, v) and (u, w) form a zero degree angle at vertex u. s_1 and s_2 are the segments of these edges that end at u. (b) We put a new dummy vertex at the end of s_2 and collapse the common part of s_1 and s_2 into a new edge.

in any drawing of H. We put a new dummy vertex at the end of s_2 and collapse the common part of s_1 and s_2 into a new edge (see Fig. 12). We apply the above algorithm for each possible choice of $(u, v), (u, w)$. At the end, we have an orthogonal representation of a 4-planar graph and can apply the algorithm in [25]. □

7.4 An Extended Branch and Bound Technique

The branch and bound technique presented in Sections 3 and 4 can be easily extended to compute simple-podevsnef drawings with the minimum number of bends of general planar biconnected graphs.

In fact, since the lower bounds presented in Section 3 can be restated in terms of simple-podevsnef drawings, they can still be used in the new drawing convention. Observe that, in this case, to compute upper and lower bounds we have to use the above network flow formulation instead of exploiting the network of [25].

There is an important point to take into account here for efficiency reasons. Consider a P-node μ_i of an SPQR-tree associated with a planar biconnected graph G and let k_i be the number of its children. Our technique requires considering all the $k_i!$ different planar embeddings of the skeleton of μ_i. If G is 4-planar, then $k_i!$ is bounded by a small constant ($k_i! \leq 6$). Else, since k_i can be linear with the number of vertices of G, we can have a combinatorial explosion of embeddings.

8 Conclusions and Open Problems

Most graph drawing problems involve the solution of optimization problems that are computationally hard. Hence, for several years, the research in graph drawing has focused on the development of efficient heuristics. However, the graphs to be drawn are frequently of reasonably small size (it is unusual to draw an Entity-Relationship or Data Flow diagrams with more than 70-80 vertices) and the available workstations allow faster and faster computation. Also, the requirements of the users in terms of aesthetic features of the interfaces are always growing.

Thus, recently there has been an increasing attention towards the development of graph drawing algorithms that privilege the effectiveness of the drawing against the efficiency of the computation (see, e.g., [19]).

In this paper, we presented an algorithm for computing an orthogonal representation with the minimum number of bends of a biconnected planar graph. The computational results that we obtained are encouraging both in terms of the quality of the drawings and in terms of the time performance.

This paper opens several problems. For example:

- Is it possible to apply other techniques, different from the branch and bound one, to tackle the problem of minimizing the number of bends of orthogonal representations?
- Are there more accurate techniques for determining lower and upper bounds of the number of bends of orthogonal representations?
- Is it possible to apply variations of the approach presented in this paper to solve other problems on planar graphs whose solution is affected from the planar embedding?
- How difficult is it to find an enumeration schema and lower bounds for the problem of computing the orthogonal representation in 3-space with the minimum number of bends?

It would also be interesting to compare the experimental results presented in this paper with the ones obtained with heuristics for bends reduction. See, for example, [24].

ACKNOWLEDGMENTS

We are grateful to Antonio Leonforte for implementing part of the system. We are also grateful to Sandra Follaro, Armando Parise, and Maurizio Patrignani for their support. Research supported in part by the ESPRIT LTR Project no. 20244-ALCOM-IT.

REFERENCES

[1] C. Batini, E. Nardelli, and R. Tamassia, "A Layout Algorithm for Data Flow Diagrams," *IEEE Trans. Software Eng.*, vol. 12, no. 4, pp. 538-546, 1986.
[2] T. Biedl and G. Kant, "A Better Heuristic for Orthogonal Graph Drawings," *Proc. Second Ann. European Symp. Algorithms,* pp. 24-35, 1994.
[3] F.J. Brandenburg, M. Himsolt, and C. Rohrer, "An Experimental Comparison of Force-Directed and Randomized Graph Drawing Algorithms," *Graph Drawing (Proc. GD '95),* F.J. Brandenburg, ed., pp. 76-87, 1996.
[4] G. Di Battista, P. Eades, R. Tamassia, and I.G. Tollis, *Graph Drawing.* Upper Saddle River, N.J.: Prentice Hall, 1999.
[5] G. Di Battista, A. Garg, G. Liotta, R. Tamassia, E. Tassinari, and F. Vargiu, "An Experimental Comparison of Four Graph Drawing Algorithms," *Computational Geometry Theory and Applications,* vol. 7, pp. 303-325, 1997.
[6] G. Di Battista, G. Liotta, and F. Vargiu, "Spirality and Optimal Orthogonal Drawings," *SIAM J. Computers,* vol. 27, no. 6, pp. 1,764-1,811, 1998.
[7] G. Di Battista and R. Tamassia, "On-Line Maintenance of Triconnected Components with SPQR-Trees," *Algorithmica,* vol. 15, pp. 302-318, 1996.
[8] G. Di Battista and R. Tamassia, "On-Line Planarity Testing," *SIAM J. Computers,* vol. 25, pp. 956-997, 1996.
[9] W. Didimo and G. Liotta, "Computing Orthogonal Drawings in a Variable Embedding Setting," *Algorithms and Computation (Proc. ISAAC '98),* K. Chwa, ed., pp. 79-88, 1998.
[10] S. Even, *Graph Algorithms.* Potomac, Md.: Computer Science Press, 1979.
[11] S. Even and G. Granot, "Grid Layouts of Block Diagrams—Bounding the Number of Bends in Each Connection," *Graph Drawing (Proc. GD '94),* R. Tamassia and I.G. Tollis, eds., pp. 64-75, 1995.
[12] U. Fößmeier and M. Kaufmann, "Drawing High Degree Graphs with Low Bend Numbers," *Graph Drawing (Proc. GD '95),* F.J. Brandenburg, ed., pp. 254-266, 1996.
[13] A. Garg and R. Tamassia, "On the Computational Complexity of Upward and Rectilinear Planarity Testing," Report CS-94-10, Computer Science Dept., Brown Univ. , Providence, R.I., 1994.
[14] A. Garg and R. Tamassia, "On the Computational Complexity of Upward and Rectilinear Planarity Testing," Submitted to *SIAM J. Computing,* 1995.
[15] A. Garg and R. Tamassia, "A New Minimum Cost Flow Algorithm with Applications to Graph Drawing," *Graph Drawing (Proc. GD '96),* S.C. North, ed., pp. 201-216, 1997.
[16] M. Himsolt, "Comparing and Evaluating Layout Algorithms within GraphEd," *J. Visual Language and Computing,* vol. 6, no. 3, pp. 255-273, 1995.
[17] J. Hopcroft and R.E. Tarjan, "Dividing a Graph into Triconnected Components," *SIAM J. Computing,* vol. 2, no. 3, pp. 135-158, 1973.
[18] S. Jones, P. Eades, A. Moran, N. Ward, G. Delott, and R. Tamassia, "A Note on Planar Graph Drawing Algorithms," Technical Report 216, Dept. of Computer Science, Univ. of Queensland, 1991.
[19] M. Jünger and P. Mutzel, "Exact and Heuristic Algorithms for 2-Layer Straightline Crossing Minimization," *Graph Drawing (Proc. GD '95),* F.J. Brandenburg, ed., pp. 337-348, 1996.
[20] Y. Liu, A. Morgana, and B. Simeone, "General Theoretical Results on Rectilinear Embeddability of Graphs," *Acta Math. Applications Sinica,* vol. 7, pp. 187-192, 1991.
[21] K. Mehlhorn and S. Näher, "LEDA: A Platform for Combinatorial and Geometric Computing," *Comm. ACM,* vol. 38, no. 1, pp. 96-102, 1995.
[22] T. Nishizeki and N. Chiba, "Planar Graphs: Theory and Algorithms," *Ann. Discrete Math.,* vol. 32, 1988.
[23] A. Papakostas and I.G. Tollis, "Algorithms for Area-Efficient Orthogonal Drawings," *Computational Geometry Theory and Applications,* vol. 9, nos. 1/2, pp. 83-110, 1998.
[24] J.M. Six, K.G. Kakoulis, and I.G. Tollis, "Refinement of Orthogonal Graph Drawings," *Graph Drawing (Proc. GD '98),* S. Whitesides, ed., pp. 302-315, 1999.
[25] R. Tamassia, "On Embedding a Graph in the Grid with the Minimum Number of Bends," *SIAM J. Computing,* vol. 16, no. 3, pp. 421-444, 1987.
[26] R. Tamassia, G. Di Battista, and C. Batini, "Automatic Graph Drawing and Readability of Diagrams," *IEEE Trans. Systems, Man, and Cybernetics,* vol. 18, no. 1, pp. 61-79, 1988.
[27] R. Tamassia and I.G. Tollis, "A Unified Approach to Visibility Representations of Planar Graphs," *Discrete Computational Geometry,* vol. 1, no. 4, pp. 321-341, 1986.
[28] R. Tamassia and I.G. Tollis, "Efficient Embedding of Planar Graphs in Linear Time," *Proc. IEEE Int'l Symp. Circuits and Systems,* pp. 495-498, 1987.

Paola Bertolazzi received the Laurea degree in electrical engineering from the Univeristy of Rome in 1974. She is Dirigente di ricerca (research director) of the Istituto di Analisi dei Sistemi e Informatica of the Consiglio Nazionale delle Ricerche. Her research interests include combinatorial optimization, design and analysis of algorithms, graph drawing, and parallel algorithms. Recently, her research interests have also been in the field of information systems design and business process reengineering.

Giuseppe Di Battista received the PhD in computer science from the University of Rome "La Sapienza" in 1990. He is currently a faculty member in the Department of Computer Science and Automation at the Third University of Rome. His research interests include graph drawing, information visualization, computational geometry, graph algorithms, computer networks, and databases. He has published about 100 papers in the above areas and has given several invited lectures worldwide. He served on program committees of international symposiums and has been editor or guest editor of international journals. He is a founding member of the steering committee for Graph Drawing. His research has been funded by the Italian National Research Council, the Esprit Program of the EC, and several industrial sponsors. He is a member of the IEEE Computer Society.

Walter Didimo received the Laurea degree in mathematics from the University of Rome "La Sapienza" in 1996 and the PhD in computer science from the same university in 2000. Currently, he has a research position at the Third Univeristy of Rome. His main research interest is in graph drawing, information visualization, and software enginnering. Others research interests include computer networks, combinatorial optimization, and computational geometry.