On-Line Convex Planarity Testing

Giuseppe Di Battista, Roberto Tamassia, and
Luca Vismara

Department of Computer Science
Brown University
Providence, Rhode Island 02912

CS-95-26
August 1995

On-Line Convex Planarity Testing*

Giuseppe Di Battista†
D.I.F.A.
Università della Basilicata
Via della Tecnica 3 - 85100 Potenza, Italy
`dibattista@iasi.rm.cnr.it`

Roberto Tamassia
Department of Computer Science
Brown University
Providence, Rhode Island 02912-1910
`rt@cs.brown.edu`

Luca Vismara
Dipartimento di Informatica e Sistemistica
Università degli Studi di Roma "La Sapienza"
Via Salaria 113 - 00198 Roma, Italy
`vismara@iasi.rm.cnr.it`

Abstract

An important class of planar straight-line drawings of graphs are the convex drawings, in which all faces are drawn as convex polygons. A graph is said to be convex planar if it admits a convex drawing. We consider the problem of testing convex planarity in a semi-dynamic environment, where a graph is subject to on-line insertions of vertices and edges. We present on-line algorithms for convex planarity testing with the following performance, where n denotes the number of vertices of the graph: convex planarity testing and insertion of vertices take $O(1)$ worst-case time, insertion of edges takes $O(\log n)$ amortized time, and the space requirement of the data structure is $O(n)$. Furthermore, we give a new combinatorial characterization of convex planar graphs.

(August 22, 1995)

*Research supported in part by the National Science Foundation under grant CCR–9423847, by the U.S. Army Research Office under grant 34990–MA–MUR, by the NATO Scientific Affairs Division under collaborative research grant 911016, by the Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo and grant 94.00023.CT07 of the Consiglio Nazionale delle Ricerche, and by the Esprit II Basic Research Action of the European Community (project ALCOM). An extended abstract of this paper was presented at the *20th International Workshop on Graph-Theoretic Concepts in Computer Science (WG '94)*, Herrsching (Munich), Germany, 1994.

†Research performed in part while this author was with the Dipartimento di Informatica e Sistemistica, Università degli Studi di Roma "La Sapienza".

1 Introduction

Planar straight-line drawings of graphs are especially interesting for their combinatorial and geometric properties. A classical result independently established by Steinitz and Rademacher [33], Wagner [40], Fary [20], and Stein [32] shows that every planar graph has a planar straight-line drawing. A grid drawing is a drawing in which the vertices have integer coordinates. Independently, de Fraysseix, Pach, and Pollack [6, 7], and Schnyder [29] have shown that every n-vertex planar graph has a planar straight-line grid drawing with $O(n^2)$ area.

An important class of planar straight-line drawings are convex drawings, in which all faces are drawn as convex polygons (see Fig. 1.a, Fig. 1.b and Fig. 2.a). Convex drawings of graphs have been extensively studied in graph theory. A graph is said to be convex planar if it admits a convex drawing. Tutte [38, 39] considered strictly-convex drawings, in which faces are strictly-convex polygons (i.e. π angles are not allowed). He shows that every triconnected planar graph is strictly-convex planar, and that a strictly-convex drawing can be constructed by solving a system of linear equations. Tutte [38, 39], Thomassen [36, 37], and Chiba, Yamanouchi, and Nishizeki [2] have given combinatorial characterizations of convex and strictly-convex planar graphs. In [2] linear time algorithms are given for convex planarity testing and for constructing convex drawings with real coordinates. Chiba, Onoguchi, and Nishizeki [1] extend the results of [2] to construct "quasi convex" drawings for graphs that are not convex planar. Kant [25, 24] presents a linear time algorithm for constructing convex drawings with integer coordinates and quadratic area. Finally, parallel algorithms for convex planarity testing and for constructing convex drawings have been given by Dehne, Djidjev and Sack [8] and by He and Kao [21].

In this paper, we present the following results on convex planarity:

- We give a new combinatorial characterization of convex planar graphs and strictly-convex planar graphs. Our characterization appears to be simpler and more intuitive than the ones previously presented in the literature [2, 36, 37, 38, 39].
- We consider the problem of testing convex planarity in a semi-dynamic environment, where a graph is subject to on-line insertions of vertices and edges. We present on-line algorithms for convex and strictly-convex planarity testing with the following performance, where n denotes the number of vertices of the graph: (strictly-) convex planarity testing and insertion of vertices take $O(1)$ worst-case time, insertion of edges takes $O(\log n)$ amortized time, and the space requirement of the data structure is $O(n)$.

Note that the (strictly-) convex planarity property is not monotone. Namely, there exists a sequence of insertions of vertices and edges on a planar graph G such that G alternatively gains or loses (strictly-) convex planarity after each insertion operation.

Previous work on dynamic planarity testing and dynamic graph drawing is reported in [3, 4, 14, 15, 16, 17, 18, 19, 26, 41].

Besides their theoretical significance, our results are motivated by the development of advanced graph drawing systems. A variety of visualization applications require to automatically draw graphs. Examples include programming environments (e.g., displaying entity-relationship diagrams and subroutine-call graphs), algorithms animation systems (e.g., representing data structures), and project planning systems (e.g., displaying PERT diagrams and organization charts). Several graph drawing systems have been recently devised (see, for example, [11, 22]). Such systems usually have a library of graph drawing algorithms, each devised to take into account a specific set of aesthetic requirements. Thus, in graph drawing systems the problem of selecting, among a set of algorithms, the one that provides the "best" visualization is of crucial importance. Since graph drawing systems are used interactively, the above selection problem has to be solved under tight performance requirements, especially for large graphs. The problem becomes harder when the graphs to be represented are subject to frequent updates. For example, given a graph, the system may need to quickly determine whether a convex drawing algorithm can be applied.

Open problems left by this work include:

- Reduce the amortized time complexity of the various operations to $O(\alpha(n))$. La Poutré [26] has recently shown that on-line planarity testing can be done within this bound.
- Develop a fully dynamic convex planarity testing algorithm. The best algorithm for fully dynamic planarity testing performs query and update operations in amortized time $O(\sqrt{n})$ [18].
- Characterize the area required by a strictly-convex drawing. Kant [25, 24] has shown that convex drawings with integer coordinates can be constructed with quadratic area. It is also known that drawing a cycle as a strictly-convex polygon with integer vertex coordinates requires cubic area [27].

The rest of this paper is organized as follows. Preliminary definitions are given in Section 2. Section 3 contains the combinatorial characterization. The on-line algorithm for convex planarity testing is presented in Section 4.

2 Preliminaries

We assume familiarity with planar graphs and with graph connectivity [28]. We recall the following definitions: a *separating k-set* of a graph is a set of k vertices whose removal disconnects the graph; separating 1-sets and 2-sets are called *cut-vertices* and *separation pairs*, respectively; a graph is *biconnected* if it is connected and has no cut-vertices, it is *triconnected* if it is biconnected and has no separation pairs.

For background on graph drawing, see [5, 9, 10, 13, 12, 34]. A *drawing* of a graph maps each vertex to a distinct point of the plane and each edge (u, v) to a simple Jordan curve with end-points u and v. A drawing is *planar* if no two edges intersect, except, possibly, at common end-points. A graph is planar if it has a planar drawing. A *straight-line* drawing is a drawing such that each edge is mapped to a straight-line segment.

Two planar drawings of a planar graph G are *equivalent* if, for each vertex v, they have the same circular clockwise sequence of edges incident with v. Hence, the planar drawings of G are partitioned into equivalence classes. Each of those classes is called an *embedding* of G. An *embedded* planar graph (also *plane* graph) is a planar graph with a prescribed embedding. A triconnected planar graph has a unique embedding, up to a reflection. A planar drawing divides the plane into topologically connected regions delimited by cycles; such cycles are called *faces*. The *external* face is the boundary of the unbounded region. Two equivalent planar drawings have the same faces. Hence, one can refer to the faces of an embedding.

A *convex* drawing of a planar graph G is a planar straight-line drawing of G in which all the faces are drawn as convex polygons. A *strictly-convex* drawing of a planar graph G is a planar straight-line drawing of G in which all the faces are drawn as strictly-convex polygons (i.e., no π angle is allowed). A graph is said *(strictly-) convex planar* if it admits a (strictly-) convex drawing.

Lemma 1 *A graph is (strictly-) convex planar only if it is biconnected.*

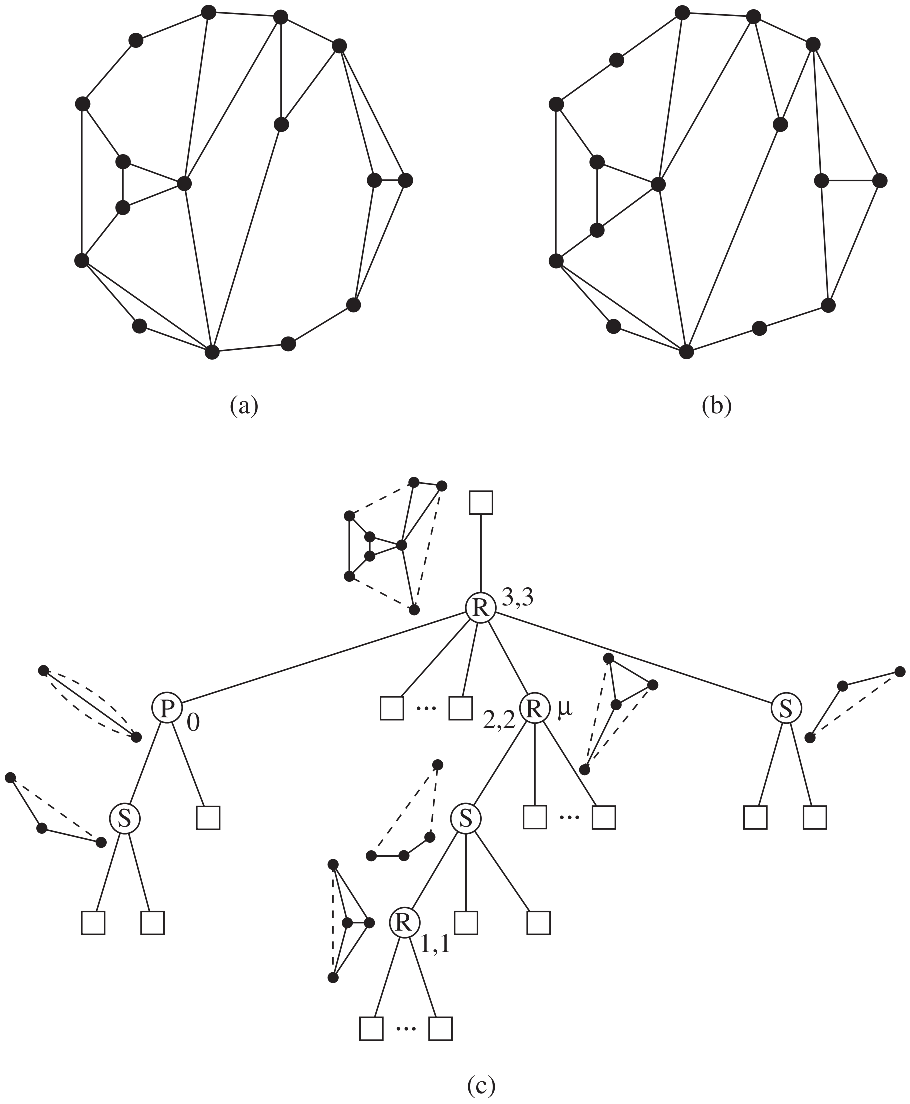
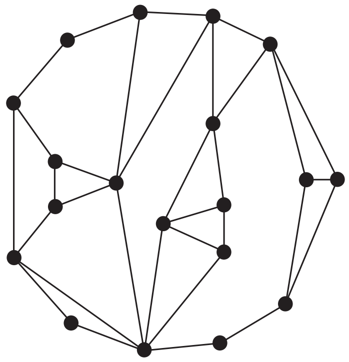
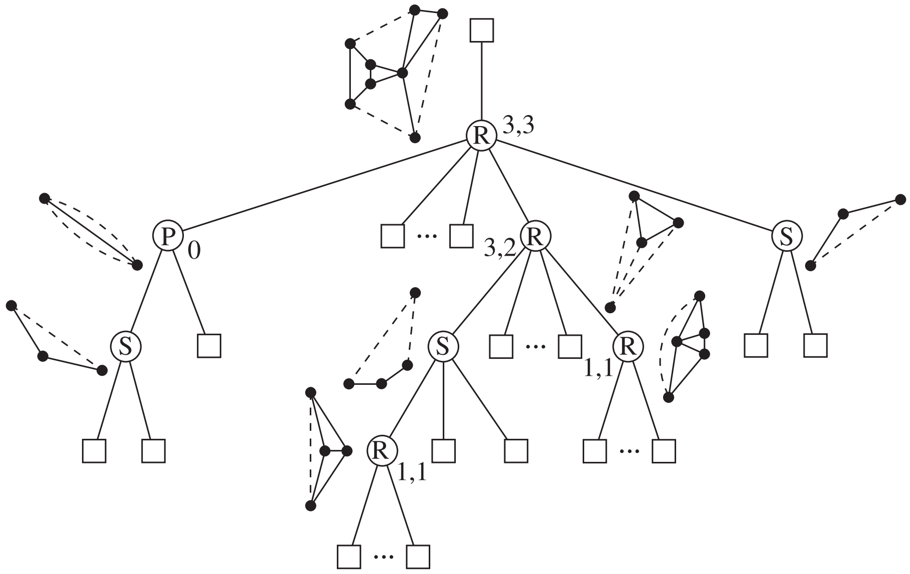


Figure 1: (a) A strictly-convex drawing of a biconnected planar graph G. (b) A convex drawing of G. (c) The SPQR-tree of G and the skeletons of its nodes. The non-trivial virtual edges are drawn with dashed lines, the trivial virtual edges are represented by solid lines, the Q-nodes are represented by squares, and the skeletons of the Q-nodes are not shown.



(a)



(b)

Figure 2: (a) A non-convex drawing of a biconnected planar graph G. (b) The SPQR-tree of G and the skeletons of its nodes.

Our data structure makes use of rooted trees. The dynamic trees of Sleator and Tarjan [30, 31] support link/cut operations and various queries (such as finding the lowest-common ancestor of two nodes) in logarithmic time. As shown in [19], they can be modified to support ordered trees and expand/contract operations. In the description of time bounds we use standard concepts of amortized complexity [35].

In the rest of this section, the SPQR-tree presented in [14, 15, 16, 17] is described. Let G be a biconnected graph. A *split pair* of G is either a pair of adjacent vertices or a separation pair. In the former case the split pair is said *trivial*, in the latter *non-trivial*. A *split component* of a split pair $\{u, v\}$ 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. In the former case the split component is said *trivial*, in the latter *non-trivial*. Note that each vertex of G distinct from u and v belongs to exactly one non-trivial split component of $\{u, v\}$. 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* T of G with respect to e describes a recursive decomposition of G induced by its split pairs. Tree T is a rooted ordered tree whose nodes are of four types: S, P, Q, and R. Each node μ of 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 T is recursively defined as follows.

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

Parallel Case: If the split pair $\{s, t\}$ has at least three split components $G_1, \cdots, G_k$ $(k \geq 3)$, the root of T is a P-node μ. Graph $skeleton(\mu)$ consists of k parallel edges between s and t, denoted $e_1, \cdots, e_k$, with $e_1 = e$.

Series Case: If the split pair $\{s, t\}$ has exactly two split components, one of them is the reference edge e, and we denote with G' the other split component. If G' has cut-vertices $c_1, \cdots, c_{k-1}$ $(k \geq 2)$ that partition G into its blocks $G_1, \cdots, G_k$, in this order from s to t, the root of 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 $(i = 1, \cdots, k)$.

Rigid Case: If none of the cases above 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 split components of $\{s_i, t_i\}$ except the one containing the reference edge e. The root of T is an R-node μ. Graph $skeleton(\mu)$ is obtained from G by replacing each subgraph G_i with the edge e_i between s_i and t_i.

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 $(i = 1, \cdots, k)$. The tree 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. Examples of SPQR-trees are shown in Fig. 1.c and in Fig. 2.b.

The *virtual edge* of node μ_i is edge e_i of $skeleton(\mu)$. A virtual edge is said *trivial* if the corresponding node μ_i is a Q-node, *non-trivial* otherwise. The endpoints of e_i are called the *poles* of μ_i. Graph G_i is called the *pertinent graph* of node μ_i, and the *expansion graph* of edge e_i.

Let μ be a node of T. We have:

- if μ is an R-node, then $skeleton(\mu)$ is a triconnected graph;
- if μ is an S-node, then $skeleton(\mu)$ is a cycle;
- if μ is a P-node, then $skeleton(\mu)$ is a triconnected multigraph consisting of a bundle of multiple edges;
- if μ is a Q-node, then $skeleton(\mu)$ is a biconnected multigraph consisting of two multiple edges.

The skeletons of the nodes of T are homeomorphic to subgraphs of G. Also, the union of the sets of split pairs of the skeletons of the nodes of T is equal to the set of split pairs of G. It is possible to show that SPQR-trees of the same graph with respect to different reference edges are isomorphic and are obtained one from the other by selecting a different Q-node as the root. SPQR-trees are closely related to the classical decomposition of biconnected graphs into triconnected components [23]. Namely, the triconnected components of a biconnected graph G are in one-to-one correspondence with the internal nodes of the SPQR-tree: the R-nodes correspond to triconnected graphs, the S-nodes to polygons, and the P-nodes to bonds. SPQR-trees of planar graphs were introduced in [14] and applied to the problem of on-line planarity testing.

The SPQR-tree T of a planar graph 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 T is $O(n)$.

3 A New Characterization of (Strictly-) Convex Planar Graphs

Let Γ be a planar straight-line drawing of a biconnected planar graph G. A vertex of G is said *external* (resp., *internal*) in Γ if it is (resp., it is not) a vertex of the external face of Γ. An *external* (resp., *internal*) edge in Γ is defined analogously. A subgraph G' of G is *drawn outside* (resp., *inside*) Γ if G' has (resp., does not have) external edges. The *external cycle* of a split component C in Γ is the cycle of G bounding the region of the plane in which C is drawn.

Lemma 2 *Let G be a biconnected planar graph and let Γ be a strictly convex drawing of G. The non-trivial split components of G are drawn outside Γ.*

Proof: Suppose, for a contradiction, that a non-trivial split component C of a split pair $\{u, v\}$ is drawn inside Γ. Vertices u and v divide the external cycle of C into two (possibly not disjoint) paths p_1 and p_2. Path p_1 (p_2) is part of an internal face f_1 (f_2) of G (see Fig. 3). By easy geometric considerations, it follows that if f_1 is drawn as a strictly-convex polygon in Γ then f_2 is not and vice versa. Thus Γ is not a strictly-convex drawing: a contradiction. □

Corollary 1 *Let G be a biconnected planar graph and let Γ be a strictly-convex drawing of G. For each separation pair $\{u, v\}$, vertices u and v must be external in Γ.*

Proof: Suppose, for a contradiction, that at least one vertex of separation pair $\{u, v\}$ is internal in Γ. All but one split components of $\{u, v\}$ are drawn inside Γ; thus, by Lemma 2, Γ is not a strictly-convex drawing: a contradiction. □

We are now ready to state the main result of this section.

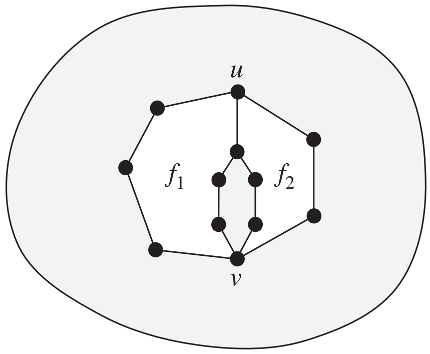


Figure 3: A split component drawn inside Γ.

Theorem 1 *Let G be a biconnected planar graph and let T be the SPQR-tree of G. Graph G is strictly-convex planar if and only if, for each node μ of T, there exists an embedding of $skeleton(\mu)$ with all the non-trivial virtual edges on the same face.*

Proof: *Only if.* Let Γ indicate a strictly-convex drawing of G.

If μ is a Q- or S-node, then $skeleton(\mu)$ is a pair of parallel edges or a cycle, respectively, and the claim is trivially true.

If μ is a P-node, then suppose, for a contradiction, that $skeleton(\mu)$ contains at least three (parallel) non-trivial virtual edges with common endpoints u and v. Even if u and v are external vertices in Γ, the expansion graph of one of the virtual edges is “drawn between” the expansion graphs of the other two, that is, drawn inside Γ. Thus, by Lemma 2, Γ is not strictly-convex: a contradiction.

If μ is an R-node, consider the expansion graphs of the non-trivial virtual edges in $skeleton(\mu)$. By Lemma 2, such expansion graphs must be drawn outside Γ. It follows that if we replace them in Γ with straight-line segments (representing their virtual edges), we obtain a planar straight-line drawing Γ_μ of $skeleton(\mu)$, in which all the non-trivial virtual edges are on the external face. The claim is then proved if we consider that $skeleton(\mu)$ is a triconnected planar graph and thus it has a unique embedding.

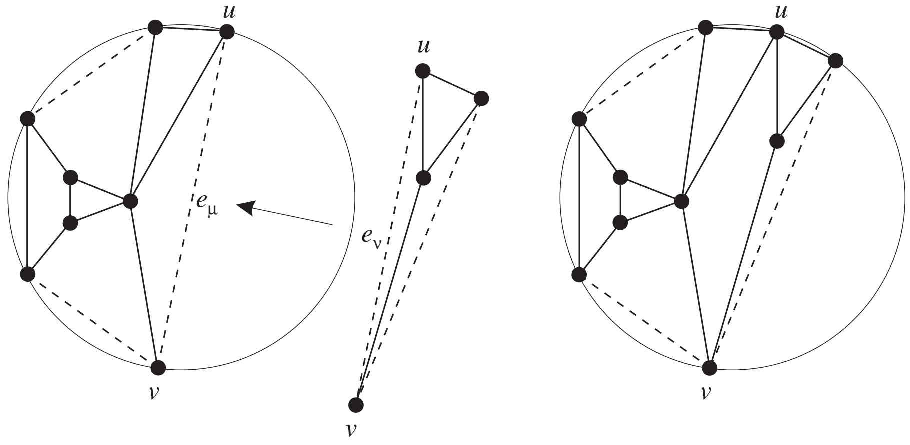
If. We show how to construct a strictly-convex drawing Γ of G in a circle c while visiting T. For each node μ of T, we choose as external the face of $skeleton(\mu)$ containing the non-trivial virtual edges and we draw $skeleton(\mu)$ in a circular segment of c.

At the beginning of the visit of T, the circular segment coincides with c and we draw the skeleton of the root of T (two parallel virtual edges, one of which trivial) as a chord of c. At each following step, let μ be the node currently visited and let ν be its parent. If μ is a P-, S-, or R- node, the virtual edge e_μ in $skeleton(\nu)$ is represented by a chord of c which identifies a circular segment s_μ (see Fig. 4.a).

If μ is a Q-node, $skeleton(\mu)$ is drawn by placing the poles of μ (i.e. the common endpoints of e_ν and of the edge of G in $skeleton(\mu)$) at the endpoints of the line segment representing e_μ.

If μ is a P-node, $skeleton(\mu)$ is drawn by placing the poles of μ (i.e. the common endpoints of e_ν and of the other two virtual edges in $skeleton(\mu)$, one of which trivial) at the endpoints of the chord identifying s_μ.

If μ is an S-node, $skeleton(\mu)$ is drawn by placing the poles of μ (i.e. the endpoints of e_ν in $skeleton(\mu)$) at the endpoints of the chord identifying s_μ, and the other vertices at distinct points of the circular arc of s_μ.



(a) (b)

Figure 4: An example of the construction in the proof of Theorem 1.

If μ is an R-node, a strictly-convex drawing of $skeleton(\mu)$ is obtained by using the algorithm of Tutte [39] or the algorithm of Chiba et al. [1, 2], with the poles of μ (i.e. the endpoints of e_ν in $skeleton(\mu)$) at the endpoints of the chord identifying s_μ and the other external vertices at distinct points of the circular arc of s_μ.

Then e_μ and e_ν are removed from the drawing. If μ is a Q-node, the whole step consists of replacing a trivial virtual edge of the drawing with an edge of G. If μ is a P-node, it consists of replacing a non-trivial virtual (external) edge of the drawing with two parallel virtual edges (one of which trivial). If μ is an S-node, it consists of appending a strictly-convex polygon to the drawing along a non-trivial virtual (external) edge, which is then removed. If μ is an R-node, it consists of appending a strictly-convex drawing of a triconnected planar graph to the drawing along a non-trivial virtual (external) edge, which is then removed (see Fig. 4.b).

Note that, at each step, the following invariants hold for the drawing that is being constructed:

- the non-trivial virtual edges are external in the drawing, and are represented by chords of c;
- if μ is an S- or R-node, the internal face generated by the removal of e_μ and e_ν is a strictly-convex polygon: in fact, the two faces sharing $e_\mu = e_\nu$ before the removal are strictly-convex polygons, the endpoints of e_μ and e_ν are placed on c, and the drawing is contained in c;
- the external face is a strictly-convex polygon, since all its vertices are on c.

The planarity of Γ follows from the planarity of the drawings of the skeletons and from the skeletons being recursively drawn inside empty circular segments of c. $\square$

It is easy to verify that the SPQR-tree of the graph in Fig. 1 satisfies the condition of Theorem 1. Hence, the graph of Fig. 1 is strictly-convex planar.

Instead, consider the SPQR-tree of the graph in Fig. 2. The skeleton of the R-node child of the root does not admit an embedding with all the non-trivial virtual edges on the same face. Hence, the condition of Theorem 1 is not satisfied, and the graph of Fig. 2 is not strictly-convex planar.

In the rest of this section we extend the characterization of Theorem 1 to non-strictly-convex drawings.

A non-trivial split component of split pair $\{u, v\}$ is said a (u, v)-*chain* if it is a path. A (u, v)-chain C is *maximal* if there is no (u', v')-chain C' such that C is a proper subpath of C'.

Lemma 3 *Let Γ be a convex drawing of a biconnected planar graph G, and let C be a (u,v)-chain of G drawn inside Γ. The following properties hold:*

1. *there exist at most three (u,v)-chains;*
2. *the (u,v)-chains distinct from C are drawn outside Γ;*
3. *u and v are not adjacent.*

Proof: We first prove that, for each split pair $\{u,v\}$, at most one (u,v)-chain C of G can be drawn inside Γ. The proof is similar to that of Lemma 2, placing the vertices of C on a straight-line segment. Properties 1, 2 and 3 easily follow. □

The *reduced* graph of a biconnected graph G is the biconnected graph G', homeomorphic to G, obtained from G in the following way. If G is a cycle, exactly one maximal (u,v)-chain is replaced with vertex w and edges (u,w), (w,v). If G is not a cycle, for each non-trivial split pair $\{u,v\}$ of G, exactly one of the maximal (u,v)-chains (if any) is replaced with edge (u,v), called *short-cut*.

Theorem 2 *A biconnected graph is convex planar if and only if its reduced graph is strictly-convex planar.*

Proof: Let G be a biconnected graph and let G' be its reduced graph. If G is a cycle the claim is trivially proved. In the rest of the proof we assume that G is not a cycle.

Only if. Let Γ_c be a convex drawing of G. As seen in the proof of Lemma 3, for each maximal (u,v)-chain C of G drawn inside Γ_c, the vertices of C are placed on a straight-line segment. By replacing each maximal (u,v)-chain of G drawn inside Γ_c with short-cut (u,v), drawn as a straight-line segment, we obtain a convex drawing Γ'_c of the reduced graph G'. Note that in Γ'_c there may still be π angles around external vertices of degree two and around vertices of degree at least three. It is easy to see that a strictly-convex drawing Γ'_{sc} of G' can always be obtained from Γ'_c by local adjustment of those vertices.

If. Let Γ'_{sc} be a strictly-convex drawing of G'. A convex drawing of G can be obtained from Γ'_{sc} by replacing each short-cut (u,v) with the corresponding maximal (u,v)-chain, drawn placing the vertices on a straight-line segment. □

4 On-Line Convex Planarity Testing

We consider a semi-dynamic environment where a planar graph G is updated by the insertion of vertices and edges that preserve planarity. The repertory of query and update operations extends the one given in [17]:

$Convex$: Determine whether G is convex planar.

$StrictlyConvex$: Determine whether G is strictly-convex planar.

$Test(v_1, v_2)$: Determine whether edge (v_1, v_2) can be added to G while preserving planarity.

$InsertEdge(e, v_1, v_2)$: Add edge e between vertices v_1 and v_2 to graph G. The operation is allowed only if the resulting graph is itself planar.

$InsertVertex(e, v, e_1, e_2)$: Split edge e into two edges e_1 and e_2 by inserting vertex v.

$AttachVertex(e, v, u)$: Add vertex v and connect it to vertex u by means of edge e.

$MakeVertex(v)$: Add an isolated vertex v.

Note that graph G may be non-biconnected (and even non-connected).

As shown in [17], $InsertEdge$, $InsertVertex$, $AttachVertex$, and $MakeVertex$ form a complete repertory of update operations for planar graphs. In particular, let G be a planar graph with n vertices. We have:

- if G is connected, G can be assembled starting from a single-vertex by means of a sequence of $O(n)$ $InsertEdge$ and $AttachVertex$ operations;
- if G is biconnected, G can be assembled starting from a three-vertex cycle by means of a sequence of $O(n)$ $InsertEdge$ and $InsertVertex$ operations.

The data structure extends the one for on-line planarity testing given in [17]. Namely, we use the following additional structures:

- For each R-node μ of T:
  - For each face f of $skeleton(\mu)$ (recall that the embedding of the skeleton of an R-node is unique), a balanced binary tree $B(f)$, where each leaf of $B(f)$ is associated with an edge e of f, and stores value 1 or 0 according to whether e is a non-trivial or trivial virtual edge, and each internal node stores the sum of the values of the leaves in its subtree (see Fig. 5). Hence, the root of $B(f)$ stores the number of non-trivial virtual edges of f, denoted $Rvirtual(f)$.
  - A balanced binary tree $B(\mu)$ associated with μ, where each leaf of $B(\mu)$ is associated with a face f of μ, and stores $Rvirtual(f)$, and each internal node stores the sum of the values of the leaves in its subtree (see Fig. 5). Hence, the root of $B(\mu)$ stores 2 times the total number of non-trivial virtual edges in $skeleton(\mu)$, this last denoted $totalRvirtual(\mu) = \frac{1}{2}\sum_f Rvirtual(f)$.
  - A variable $maxRvirtual(\mu) = \max_f\{Rvirtual(f)\}$, storing the maximum value of $virtual(f)$ over all faces f of $skeleton(\mu)$.
- For each P-node μ of T:
  - An indicator $Pthree(\mu)$, where $Pthree(\mu) = 1$ if $skeleton(\mu)$ has three or more non-trivial virtual edges, and $Pthree(\mu) = 0$ otherwise.
- Variables storing the following values:
  - The total number of non-trivial virtual edges in the skeletons of the R-nodes of T, denoted
  $$sumtotalRvirtual(G) = \sum_{\mathrm{R-node}\mu\in T} totalRvirtual(\mu);$$
  - The sum of the values $maxRvirtual(\mu)$ over all the R-node of T, denoted
  $$summaxRvirtual(G) = \sum_{\mathrm{R-node}\mu\in T} maxRvirtual(\mu);$$

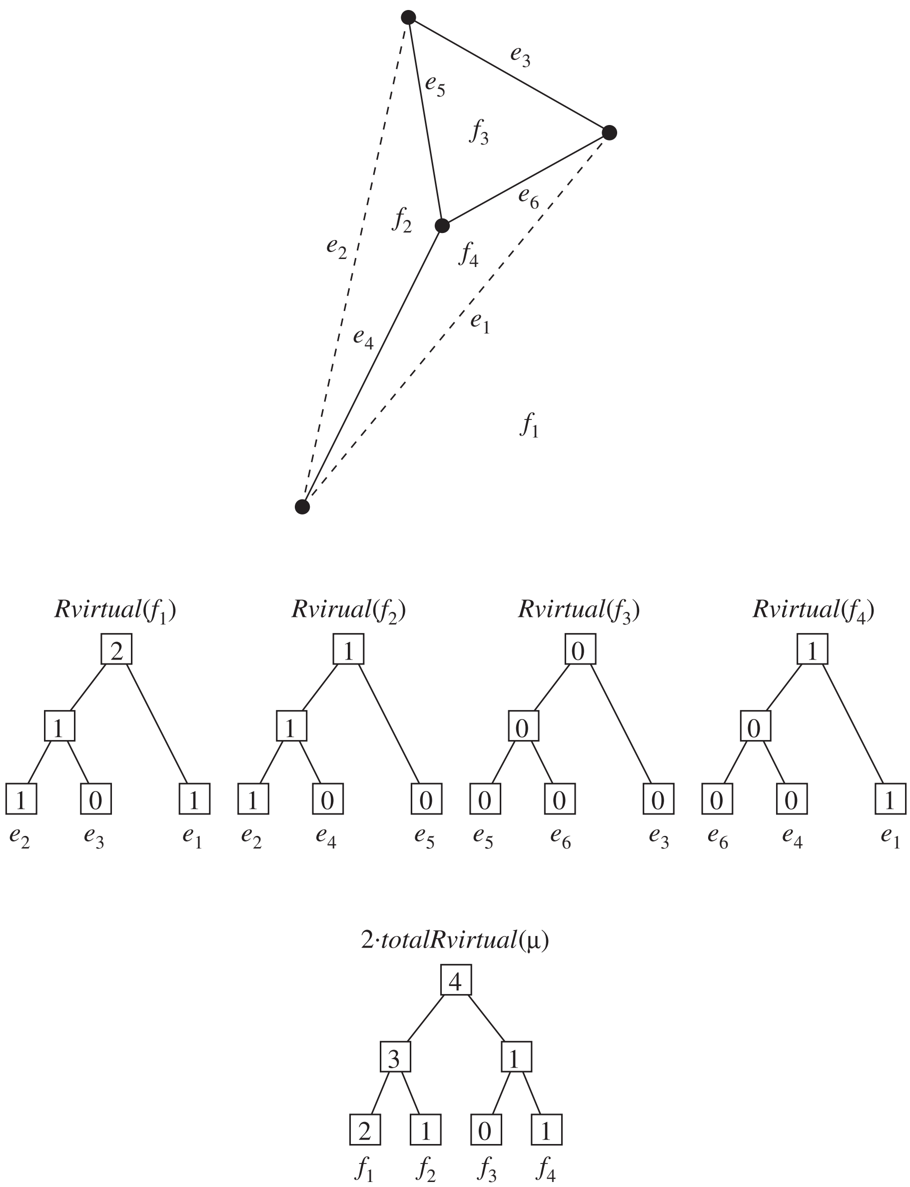


Figure 5: The skeleton of R-node μ in Fig. 1.c, the balanced binary trees for its faces, and the balanced binary tree for μ.

– The number of P-nodes of T with at least three non-trivial virtual edges, denoted

$$sumPthree(G) = \sum_{\mathrm{P-node}\mu \in T} Pthree(\mu);$$

– The total number of biconnected components of G, denoted $totalbico(G)$.

In Fig. 1.c and in Fig. 2.b, for each R-node μ of the SPQR-tree, the values of $totalRvirtual(\mu)$ and $maxRvirtual(\mu)$ are indicated (separated by a comma); likewise, for each P-node μ the value of $Pthree(\mu)$ is indicated. In Fig. 1.c, for graph G, $sumtotalRvirtual(G) = 6$, $summaxRvirtual(G) = 6$, and $sumPthree(G) = 0$. In Fig. 2.b, for graph G, $sumtotalRvirtual(G) = 8$, $summaxRvirtual(G) = 7$, and $sumPthree(G) = 0$.

Theorem 3 *Operation StrictlyConvex returns true if and only if the following conditions hold:*

1. $totalbico(G) = 1$;
2. $sumPthree(G) = 0$;
3. $sumtotalRvirtual(G) = summaxRvirtual(G)$.

Proof: Condition 1 expresses the fact that G is biconnected (see Lemma 1).

Condition 2 expresses the fact that every P-node of T has at most two non-trivial virtual edges, i.e., every split pair has at most two non-trivial split components.

For each R-node μ of T, $totalRvirtual(\mu) \geq maxRvirtual(\mu)$, where equality holds if and only if all the non-trivial virtual edges of $skeleton(\mu)$ are on the same face. It follows that, for G, $sumtotalRvirtual(G) \geq summaxRvirtual(G)$. Condition 3 holds if and only if $totalRvirtual(\mu) = maxRvirtual(\mu)$ for each R-node μ: necessity can be proved by contradiction, sufficiency is trivial. It follows that Condition 3 expresses the fact that, for each R-node μ of T, all the non-trivial virtual edges of $skeleton(\mu)$ are on the same face.

Thus, Conditions 1, 2, and 3 are equivalent to Theorem 1. □

Theorem 4 *Let G be a planar graph that is updated on-line by adding vertices and edges, and let n be the current number of vertices of G. There exists a data structure for on-line convex planarity testing of G with the following performance: the space requirement is $O(n)$; operations $MakeVertex$, $Convex$ and $StrictlyConvex$ take $O(1)$ worst-case time; operations $Test$, $AttachVertex$ and $InsertVertex$ take $O(\log n)$ worst-case time; operation $InsertEdge$ takes $O(\log n)$ amortized time.*

Proof: Let an *elementary skeleton operation* be one of the following operations:

- creation of a new skeleton with $O(1)$ vertices;
- an $InsertVertex$ or $InsertEdge$ operation performed on an existing skeleton;
- changing the type of a skeleton edge from trivial to non-trivial.

Updating the additional data structure for on-line convex planarity testing after an elementary skeleton operation can be done in $O(\log n)$ time. E.g., an $InsertVertex$ or $InsertEdge$ operation on $skeleton(\mu)$ corresponds to performing $O(1)$ insert/delete and split/splice operations on the balanced tree $B(\mu)$ and on the balanced trees $B(f)$ associated with the faces of $skeleton(\mu)$.

To analyze the time complexity of the query and update operations on G, we recall the following properties of the basic on-line planarity testing data structure [17]:

- Operations $AttachVertex$ and $InsertVertex$ take $O(\log n)$ worst-case time and require $O(1)$ worst-case elementary skeleton operations.
- Operation $InsertEdge$ takes $O(\log n)$ amortized time and requires $O(1)$ amortized elementary skeleton operations.
- The number of biconnected components of G can be maintained in $O(\log n)$ amortized time per update operation.

The above properties imply the claimed time complexity bounds.

$\square$

References

[1] N. Chiba, K. Onoguchi, and T. Nishizeki. Drawing planar graphs nicely. *Acta Inform.*, 22:187–201, 1985.

[2] N. Chiba, T. Yamanouchi, and T. Nishizeki. Linear algorithms for convex drawings of planar graphs. In J. A. Bondy and U. S. R. Murty, editors, *Progress in Graph Theory*, pages 153–173. Academic Press, New York, NY, 1984.

[3] R. F. Cohen, G. Di Battista, R. Tamassia, and I. G. Tollis. Dynamic graph drawing: Trees, series-parallel digraphs, and planar *st*-digraphs. *SIAM J. Comput.*, 24(5), 1995.

[4] R. F. Cohen, G. Di Battista, R. Tamassia, I. G. Tollis, and P. Bertolazzi. A framework for dynamic graph drawing. In *Proc. 8th Annu. ACM Sympos. Comput. Geom.*, pages 261–270, 1992.

[5] I. F. Cruz and P. Eades, editors. *Special Issue on Graph Visualization*, volume 6(3) of *J. Visual Languages and Computing*. 1995.

[6] H. de Fraysseix, J. Pach, and R. Pollack. Small sets supporting Fary embeddings of planar graphs. In *Proc. 20th Annu. ACM Sympos. Theory Comput.*, pages 426–433, 1988.

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

[8] F. Dehne, H. Djidjev, and J.-R. Sack. An optimal PRAM algorithm for planar convex embedding. In G. Di Battista, P. Eades, H. de Fraysseix, P. Rosenstiehl, and R. Tamassia, editors, *Graph Drawing '93 (Proc. ALCOM Workshop on Graph Drawing)*, pages 75–77, 1993.

[9] G. Di Battista, P. Eades, H. de Fraysseix, P. Rosenstiehl, and R. Tamassia, editors. *Graph Drawing '93 (Proc. ALCOM Workshop on Graph Drawing)*. 1993.

[10] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. Algorithms for drawing graphs: an annotated bibliography. *Comput. Geom. Theory Appl.*, 4:235–282, 1994.

[11] G. Di Battista, G. Liotta, and F. Vargiu. Diagram Server. *J. Visual Languages and Computing* (special issue on Graph Visualization, edited by I. F. Cruz and P. Eades), 6(3), 1995.

[12] G. Di Battista and R. Tamassia, editors. *Special Issue on Geometric Representations of Graphs.* Comput. Geom. Theory Appl. To appear.

[13] G. Di Battista and R. Tamassia, editors. *Special Issue on Graph Drawing.* Algorithmica. To appear.

[14] G. Di Battista and R. Tamassia. Incremental planarity testing. In *Proc. 30th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 436–441, 1989.

[15] G. Di Battista and R. Tamassia. On-line graph algorithms with SPQR-trees. In *Automata, Languages and Programming (Proc. 17th ICALP)*, volume 442 of *Lecture Notes in Computer Science*, pages 598–611, 1990.

[16] G. Di Battista and R. Tamassia. On-line maintenance of triconnected components with SPQR-trees. *Algorithmica*, to appear. Preprint: Technical Report CS-92-40, Comput. Sci. Dept., Brown Univ. (1992).

[17] G. Di Battista and R. Tamassia. On-line planarity testing. *SIAM J. Comput.*, to appear. Preprint: Technical Report CS-92-39, Comput. Sci. Dept., Brown Univ. (1992).

[18] D. Eppstein, Z. Galil, G. Italiano, and T. Spencer. Separator based sparsification for dynamic planar graph algorithms. In *Proc. 25th Annu. ACM Sympos. Theory Comput.*, pages 208–217, 1993.

[19] D. Eppstein, G. F. Italiano, R. Tamassia, R. E. Tarjan, J. Westbrook, and M. Yung. Maintenance of a minimum spanning forest in a dynamic planar graph. *J. Algorithms*, 13:33–54, 1992.

[20] I. Fary. On straight lines representation of planar graphs. *Acta Sci. Math. Szeged.*, 11:229–233, 1948.

[21] X. He and M.-Y. Kao. Parallel construction of canonical ordering and convex drawing of triconnected planar graphs. In *Proc. 4th Annu. Internat. Sympos. Algorithms Comput. (ISAAC 93)*, volume 762 of *Lecture Notes in Computer Science*, pages 303–312. Springer-Verlag, 1993.

[22] M. Himsolt. GraphEd: a graphical platform for the implementation of graph algorithms. In R. Tamassia and I. G. Tollis, editors, *Graph Drawing (Proc. GD '94)*, volume 894 of *Lecture Notes in Computer Science*, pages 182–193. Springer-Verlag, 1995.

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

[24] G. Kant. Drawing planar graphs using the canonical ordering. *Algorithmica* (special issue on Graph Drawing, edited by G. Di Battista and R. Tamassia). To appear.

[25] G. Kant. Drawing planar graphs using the *lmc*-ordering. In *Proc. 33th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 101–110, 1992.

[26] J. A. La Poutré. Alpha-algorithms for incremental planarity testing. In *Proc. 26th Annu. ACM Sympos. Theory Comput.*, pages 706–715, 1994.

[27] Y. Lin and S. Skiena. Complexity aspects of visibility graphs. Technical Report 92/08, SUNY Stony Brook, 1992.

[28] T. Nishizeki and N. Chiba. Planar graphs: Theory and algorithms. *Annals of Discrete Mathematics*, 32, 1988.

[29] W. Schnyder. Embedding planar graphs on the grid. In *Proc. 1st ACM-SIAM Sympos. Discrete Algorithms*, pages 138–148, 1990.

[30] D. D. Sleator and R. E. Tarjan. A data structure for dynamic trees. *J. Comput. System Sci.*, 26(3):362–381, 1983.

[31] D. D. Sleator and R. E. Tarjan. Self-adjusting binary search trees. *J. ACM*, 32(3):652–686, 1985.

[32] S. K. Stein. Convex maps. *Proc. Amer. Math. Soc.*, 2:464–466, 1951.

[33] E. Steinitz and H. Rademacher. *Vorlesungen über die Theorie der Polyeder.* Julius Springer, Berlin, Germany, 1934.

[34] R. Tamassia and I. G. Tollis, editors. *Graph Drawing (Proc. GD '94)*, volume 894 of *Lecture Notes in Computer Science*. Springer-Verlag, 1995.

[35] R. E. Tarjan. Amortized computational complexity. *SIAM J. Algebraic Discrete Methods*, 6(2):306–318, 1985.

[36] C. Thomassen. Planarity and duality of finite and infinite planar graphs. *J. Combin. Theory Ser. B*, 29:244–271, 1980.

[37] C. Thomassen. Plane representations of graphs. In J. A. Bondy and U. S. R. Murty, editors, *Progress in Graph Theory*, pages 43–69. Academic Press, New York, NY, 1984.

[38] W. T. Tutte. Convex representations of graphs. *Proceedings London Mathematical Society*, 10(3):304–320, 1960.

[39] W. T. Tutte. How to draw a graph. *Proceedings London Mathematical Society*, 13(3):743–768, 1963.

[40] K. Wagner. Bemerkungen zum vierfarbenproblem. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 46:26–32, 1936.

[41] J. Westbrook. Fast incremental planarity testing. In *Automata, Languages and Programming (Proc. 19th ICALP)*, volume 623 of *Lecture Notes in Computer Science*, pages 342–353, 1992.