# Convex Drawings of Hierarchical Graphs in Linear Time, with Applications to Planar Graph Morphing[*]

**Boris Klemz**
Universität Würzburg, Germany

**Abstract**

A hierarchical plane st-graph $H$ can be thought of as a combinatorial description of a planar drawing $\Gamma$ of a 2-connected graph $G$ in which each edge is a $y$-monotone curve and each face encloses a $y$-monotone region (that is, a region whose intersection with any horizontal line is a line segment, a point, or empty). A drawing $\Gamma'$ of $H$ is a drawing of $G$ such that each horizontal line intersects the same left-to-right order of edges and vertices in $\Gamma$ and $\Gamma'$, that is, the underlying hierarchical plane st-graph of both drawings is $H$. A straight-line planar drawing of a graph is convex if the boundary of each face is realized as a convex polygon.

We study the computation of convex drawings of hierarchical plane st-graphs such that the outer face is realized as a prescribed polygon. Chrobak, Goodrich, and Tamassia [SoCG'96] and, independently, Kleist et al. [CGTA'19] described an idea to solve this problem in $\mathcal{O}(n^{1.1865})$ time, where $n$ is the number of vertices of the graph. Also independently, Hong and Nagamochi [J. Discrete Algorithms'10] described a completely different approach, which can be executed in $\mathcal{O}(n^2)$ time.

In this paper, we present an optimal $\mathcal{O}(n)$ time algorithm to solve the above problem, thereby improving the previous results by Chrobak, Goodrich, and Tamassia, Kleist et al., and by Hong and Nagamochi. Our result has applications in graph morphing. A planar morph is a continuous deformation of a graph drawing that preserves straight-line planarity. We show that our algorithm can be used as a drop-in replacement to speed up a procedure by Alamdari et al. [SICOMP'17] to morph between any two given straight-line planar drawings of the same plane graph. The running time improves from $\mathcal{O}(n^{2.1865})$ to $\mathcal{O}(n^2 \log n)$. To obtain our results, we devise a new strategy for computing so-called archfree paths in hierarchical plane st-graphs, which might be of independent interest.

**2012 ACM Subject Classification** Mathematics of computing → Paths and connectivity problems; Mathematics of computing → Graphs and surfaces; Theory of computation → Computational geometry; Human-centered computing → Graph drawings

**Keywords and phrases** convex drawing, hierarchical graph, graph drawing, computational geometry, planarity, planar graph, morphing, convexity

**Digital Object Identifier** 10.4230/LIPIcs.ESA.2021.57

**Funding** *Boris Klemz*: supported by DFG project WO 758/11-1.

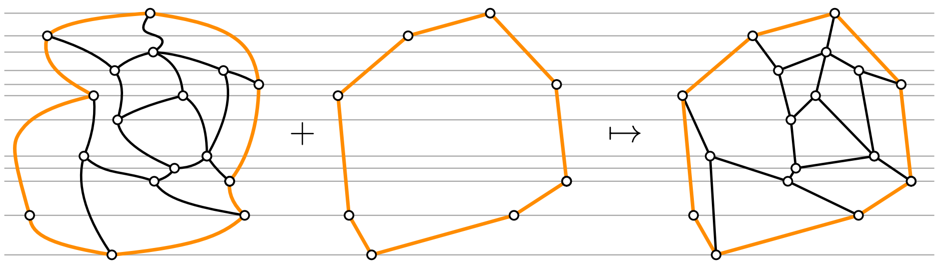


**Figure 1** A $y$-monotone drawing (left) and a convex drawing of its underlying hierarchical plane st-graph (right) with a prescribed compatible polygon (middle) as the outer face.

[*] Due to space constraints, some proofs in this extended abstract are only sketched or omitted entirely.

## 1 Introduction

A *$y$-monotone* drawing $\Gamma$ is a planar drawing of a 2-connected planar graph $G = (V, E)$ where each edge is realized as a $y$-monotone curve and the boundary of each face encloses a *$y$-monotone* region, that is, a region whose intersection with any horizontal line is a line segment, a point, or empty; for an illustration see Figure 1. Due to the $y$-monotone edges, the drawing $\Gamma$ uniquely determines for each $v \in V$ (1) the $y$-coordinate $\mathrm{y}(v)$ of $v$, (2) a left-to-right ordering of the edges incident to $v$ that have an endpoint with a $y$-coordinate larger than $\mathrm{y}(v)$, and (3) a left-to-right ordering of the edges incident to $v$ that have an endpoint with a $y$-coordinate smaller than $\mathrm{y}(v)$. A plane graph is a combinatorial description of a planar drawing of a graph that consists of the graph equipped with the so-called combinatorial embedding of the drawing and a distinguished outer face. Similarly, the underlying *hierarchical plane st-graph* $H$ of $\Gamma$ is a combinatorial description of the $y$-monotone drawing $\Gamma$ that consists of $G$ equipped with the above information (Items (1)–(3)) for each vertex. A *drawing* $\Gamma'$ of $H$ is a $y$-monotone drawing of $G$ whose underlying hierarchical plane st-graph is $H$. A planar straight-line drawing of a graph is called *convex* if the boundary of each face is realized as a simple convex polygon.

This paper is concerned with the computation of convex drawings of hierarchical plane st-graphs such that the outer face is realized as a prescribed polygon that is, in some sense, compatible with the given graph; for an illustration see Figure 1. A plane graph admits a convex drawing if and only if it is a subdivision of an internally 3-connected graph. Hence, we will assume that our input graphs satisfy this property.

The above problem has applications in graph morphing, as we will discuss next.

**Applications.** A *(planar) morph* is a continuous deformation of a graph drawing that preserves straight-line edges (and planarity) at all times. Graph morphing is motivated by applications in animation and computer graphics [17]. In computational morphing problems, one typically seeks "piece-wise linear" morphs, which are composed of a number of linear morphing steps. In a *linear* morph, each vertex moves along a line segment at constant speed (which depends on the length of the segment) such that it arrives at its final position at the end of the morph. Such a morph is uniquely defined by specifying the initial and the final drawing. Hence, a morph composed of $k$ linear morphs can be efficiently encoded as a sequence of $k + 1$ drawings.

Algorithms to compute convex drawings of hierarchical plane st-graphs such that the outer face is realized as a prescribed polygon serve as a subroutine in several graph morphing algorithms [2, 22, 7, 27, 6]. The key observation is that the linear morph from a $y$-monotone straight-line drawing to a convex drawing of its underlying hierarchical plane st-graph is planar, which is not difficult to prove due to the fact that its vertex trajectories are parallel lines [2]; a linear morph with this property is called *unidirectional*. The restriction to $y$-monotone drawings might seem limiting at first. However, the observation is also useful in a more general context: let $\Gamma$ be a straight-line planar drawing. Let $\Gamma'$ be a $y$-monotone augmentation of $\Gamma$ created by adding a set $A$ of $y$-monotone (but not necessarily straight-line) edges. Finally, let $\Gamma''$ be a convex drawing of the underlying hierarchical plane st-graph of $\Gamma'$. Then the linear morph from $\Gamma$ to $\Gamma'' \setminus A$ is also planar [22]. Note that all angles of face boundaries in $\Gamma'' \setminus A$ that were not subdivided by edges of $A$ are convex. In this sense, $\Gamma'' \setminus A$ is a simplified version of $\Gamma$, which can be exploited algorithmically. We will illustrate this by discussing an explicit example.

One of the most basic tasks in graph morphing is the computation of a planar morph (composed of linear morphing steps) between two given straight-line planar drawings $\Gamma_1, \Gamma_2$ of the same plane graph $G$. Alamdari et al. [2] described an algorithm to compute such a morph consisting of a linear number of unidirectional steps. They provide a reduction to show that it suffices to consider the case that $\Gamma_1, \Gamma_2$, and $G$ are triangulated. For triangulations, the idea is as follows: find a suitable edge $e = \{u, v\}$ shrink it to a point in both drawings, thereby contracting the edge $e$ in $G$ to a new vertex $w$. Then, recursively compute a morph $\mathcal{M}$ of the reduced graph. Finally, turn the thereby obtained "pseudomorph" of $G$ into an actual morph by placing $u$ and $v$ very close to the position of $w$ in each of the drawings encoding $\mathcal{M}$. To find an edge $e$ that can be contracted without violating planarity in either drawing, proceed as follows: let $u$ be an internal vertex with $\deg_G(u) \leq 5$. If the polygon defined by the neighbors of $u$ is convex in both $\Gamma_1$ and $\Gamma_2$, then $u$ can be contracted towards the same neighbor in both drawings. If this is not the case, the polygon (or some specific angle) can be made convex using the strategy described in the preceding paragraph.

In other types of morphing problems, the task is to compute planar morphs between two given drawings while maintaining additional properties such as convexity [7] or upward-planarity [27], or planar morphs that transform a given drawing in order to achieve a certain property while being in some sense monotonic [1, 22, 11]. Very recently, the problem of morphing graphs was also studied on the torus [9].

We remark that the computation of convex drawings of hierarchical plane st-graphs with a prescribed polygon as the outer face also plays a role when embedding polyhedral graphs in $\mathbb{R}^3$ with a good vertex resolution [10, 29]. The idea is to first find a convex drawing in the plane where the vertices are placed at, suitably chosen, prescribed small integer $y$-coordinates. The drawing is then lifted to $\mathbb{R}^3$. However, to ensure this strategy can be carried out, the constructed plane drawings are also required to be embeddings with so-called equilibrium stress – the drawings created by the algorithm presented in this paper do not guarantee this property.

**Previous algorithms.** An idea for constructing convex drawings of hierarchical plane st-graphs with a prescribed polygon as the outer face was described already in 1996 by Chrobak, Goodrich, and Tamassia [10, Section 3] (in the context of realizing polyhedral graphs in $\mathbb{R}^3$); also see [29, Section 4]. The approach was independently rediscovered by Kleist et al. [22] (in the context of a morphing problem) and is based on using Tutte's well-known spring theorem [32, 15, 16]. The main idea is to precompute barycenter weights that force the vertices to lie at the prescribed $y$-coordinates before applying Tutte's algorithm, which then finds suitable $x$-coordinates. (Notably, the way in which the barycenter weights are determined in [22] is quite different from the method used in [10] and [29].) Chrobak et al. point out that the approach can be implemented in $\mathcal{O}(n^{\omega/2} + n \log n) \subseteq \mathcal{O}(n^{1.1865})$ time by using the generalized nested dissection method [25, 26, 4], where $n$ is the number of vertices of the graph and $\omega < 2.37286$ [3, 24] denotes the matrix multiplication exponent.

In 2010, Hong and Nagamochi [20] described a completely different approach, based on a recursive combinatorial construction. The runtime of their algorithm is $\mathcal{O}(n^2)$. The idea is to choose a suitable internal vertex $y$ of the given graph $G$ and compute three disjoint (except for $y$) paths from $y$ to the outer face, see Figure 2a. These paths dissect $G$ into three regions, which are then handled recursively. The outer face of each of the three regions is composed of two of the three paths, which are prescribed to be realized as straight-line segments, and a part of the original prescribed polygon, see Figure 2b. To ensure that the thereby described polygon can be extended to a convex drawing of the entire region, the computed paths need to be *archfree*, meaning that they are not *arched* by an internal face. A path $P$ is arched by a face $f$ if $P$ contains two distinct vertices $a, b$ that belong to the boundary of $f$ such that the subpath $P_{ab}$ of $P$ between $a$ and $b$ is not a subpath of the boundary of $f$, see Figure 2c. Indeed, such a path $P$ cannot be realized as a straight-line segment in a convex drawing since in this case the interior of the segment $ab$ has to be disjoint from the realization of $f$.

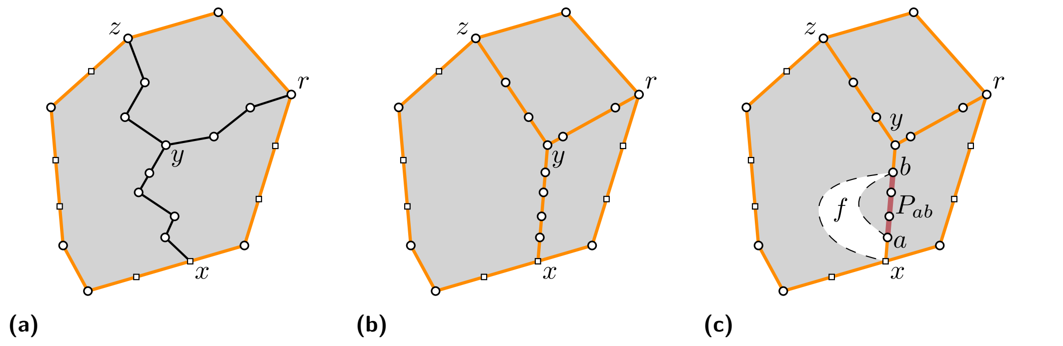


**Figure 2** (a–b) The idea of Hong and Nagamochi's [20] construction. (c) The path $P = (x, \dots, y)$ is arched by a face $f$.

We remark that Chrobak et al.'s result [10] does not appear to be widely known: the morphing papers [2, 7, 27, 6, 22] only refer to the method by Hong and Nagamochi [20]. Moreover, Hong and Nagamochi [20] and Kleist et al. [22] were also unaware of its existence.

**Other related work.** Pach and Tóth [28] and Eades, Feng, Lin, and Nagamochi [14] studied the problem of finding (not necessarily convex) straight-line drawings of *hierarchical plane graphs*, which can be defined as hierarchical plane *st*-graphs, except that they describe (not necessarily $y$-monotone) planar drawings in which each edge is realized as a $y$-monotone curve. Eades et al. [14] provide a linear-time algorithm for this problem that realizes the outer face as a prescribed polygon.

A graph in which each vertex is equipped with a $y$-coordinate is sometimes called a *level* graph. The central question in the Level Planarity [13, 18, 21] problem and its many variants [5, 8, 23] is to decide whether a given level graph admits a *level* planar drawing, that is, a planar drawing in which each vertex is placed at its prescribed $y$-coordinate and each edge is realized as a $y$-monotone curve. By definition, the underlying level graph of each hierarchical plane (st-)graph admits a level planar drawing.

**Contribution and organization.** In Section 4, we describe an optimal linear-time algorithm for constructing convex drawings of hierarchical plane st-graphs with a prescribed polygon as the outer face (if possible), thereby improving the previous approaches by Chrobak, Goodrich, and Tamassia [10], Kleist et al. [22], and Hong and Nagamochi [20].

▶ **Theorem 1.** *There exists an algorithm that, given a subdivision $G = (V, E)$ of an internally 3-connected hierarchical plane st-graph and a convex polygon $\Gamma^o$ that is compatible with $G$, computes a convex drawing of $G$ with $\Gamma^o$ as the realization of the outer face in time $\mathcal{O}(n)$ where $n = |V|$.*

We introduce our notation and terminology (including the definitions of compatible polygons and internally 3-connected graphs) and the assumed data structures in Section 2.

To obtain Theorem 1, we follow the idea of the recursive combinatorial construction by Hong and Nagamochi [20]. We observe that the main bottleneck in Hong and Nagamochi's algorithm is the computation of archfree paths to the boundary: Hong and Nagamochi obtain such a path by computing an arbitrary $y$-monotone path $P$ which is then modified to obtain an archfree path $P'$. In general, not all vertices of $P$ belong to $P'$. Hence, a given vertex may be involved in the archfree path computation on multiple layers of the recursion. Since the recursion depth can be linear, the overall runtime of their algorithm is quadratic. In Section 3, we describe a new efficient algorithm to obtain archfree paths in a more direct way: the running time of our approach is linear in the sum of the degrees of the internal vertices visited by the computed path. With this tool at hand, the layers of the recursion become, in a sense, disjoint (since archfree paths computed on distinct layers of the recursion are disjoint). This lets us carry out the algorithm corresponding to Theorem 1 in linear time.

▶ **Theorem 2.** *There exists an algorithm that, given an internally* 3*-connected hierarchical plane st-graph $G$ and (a pointer to) an internal vertex $y$ of $G$, computes an archfree directed path $P_{yz}$ from $y$ to some outer vertex $z$ of $G$ such that all vertices of $P_{yz}$ except for $z$ are internal vertices of $G$ in time $\mathcal{O}(\sum_{i \in \mathrm{V}(P_{yz}) \setminus \{z\}} \deg^+_G(i))$.*

Alamdari et al. state in [2] that their morphing algorithm, which was already mentioned above, can be executed in time $\mathcal{O}(n^3)$. It uses Hong and Nagamochi's [20] algorithm for constructing convex drawings of hierarchical plane st-graphs as a blackbox with running time $\mathcal{O}(n^2)$. Kleist et al. [22] observed that by replacing the contents of this blackbox with an algorithm based on Tutte's theorem (as described by Chrobak et al. [10]), the running time can be improved to $\mathcal{O}(n^{1+\omega/2} + n^2 \log n) \subseteq \mathcal{O}(n^{2.1865})$. By plugging in Theorem 1 instead, we further improve the running time to $\mathcal{O}(n^2 \log n)$. In fact, if the graph is 2-connected, the running time can be improved to $\mathcal{O}(n^2)$. [2, Theorem 1.1] and [22, Theorem 2] become:

▶ **Theorem 3.** *There exists an algorithm that, given two straight-line planar drawings of the same $n$-vertex plane graph $G$, computes a planar morph between the two drawings that consists of $\mathcal{O}(n)$ unidirectional morphs in time $\mathcal{O}(n^2 \log n)$, and in time $\mathcal{O}(n^2)$ if $G$ is* 2*-connected.*

Alamdari et al. [2] proved that the number of linear morphing steps to morph between two planar straight-line drawings of a path is bounded by $\Omega(n)$. This bound easily extends to the 2-connected case [22, Theorem 3]. Recall that a linear morph is uniquely defined by the initial and the final drawing. Hence, a natural way to encode a morph composed of $k$ linear morphing steps is to provide a list of $k+1$ drawings. Since the size of each drawing is $\Omega(n)$, the $\Omega(n)$ bound for the number of morphing steps implies an output complexity of $\Omega(n^2)$ when the morph is assumed to be encoded in this fashion. Hence, in this model, Theorem 3 is near-optimal, and optimal in the 2-connected case. It is an interesting question whether allowing some sort of implicit encoding can lead to better running times.

It seems likely that a similar running time improvement can be obtained for other morphing algorithms (such as the ones stated in [22, 7, 27]), though, we have not analyzed the respective running times in detail yet. We believe that Theorem 3 might be a useful tool for designing morphing algorithms in the future.

## 2 Terminology, data structures, and preliminary results

All graphs in this paper are simple, that is, we do not allow parallel edges or self-loops. Let $G = (V, E)$ be a graph. We denote by $\mathrm{V}(G) = V$ the vertex set and by $\mathrm{E}(G) = E$ the edge set of $G$. Assume that $G$ is planar and let $\Gamma$ be a planar drawing of $G$. The boundary of each face $f$ of $G$ can be uniquely described by a counterclockwise sequence of edges, or multiple such sequences if $G$ is disconnected. In the connected case, we use $\partial f$ to denote the boundary of $f$. If $G$ is 2-connected, then $\partial f$ is a *simple* cycle (otherwise, $\partial f$ can visit vertices and edges multiple times). The drawing $\Gamma$ determines a circular ordering of the neighbors of each vertex. The set of these orderings together with the set of face boundaries is called the *combinatorial embedding* of $\Gamma$. Two drawings of $G$ may have the same combinatorial embedding, but different outer faces. A *plane* graph is a planar graph equipped with a combinatorial embedding and a distinguished outer face. A plane graph can be efficiently encoded and traversed by means of the well-known *doubly connected edge list* (DCEL) [12].

Let $H$ be a hierarchical plane st-graph. We consider each edge of $H$ to be directed from its lower to its higher endpoint. Note that $H$ has a unique source and sink, which belong to the outer face. Moreover, the boundary of each face of $H$ has a unique source and sink. We refer to this as the *st-property* of $H$. For a face $f$ of $H$, we define its *peak*, denoted by $\mathrm{peak}(f)$, to be the $y$-coordinate of its sink. The *valley* of $f$, denoted by $\mathrm{valley}(f)$, is the $y$-coordinate of its source. Let $v \in \mathrm{V}(H)$ such that $v$ is not the source or sink of $H$. There is a natural partition of the faces incident to $v$: a face with two out-edges of $v$ on its boundary is called an *up-face* of $v$. Similarly, a face with two in-edges of $v$ on its boundary is called a *down-face* of $v$. The unique *left-face* of $v$ has the left-most out-edge and the left-most in-edge of $v$ on its boundary. The *right-face* of $v$ is defined analogously. Moreover, we refer to the collection of the left-, right-, and up-faces of $v$ as its *out-faces*, and to the collection of the left-, right-, and down-faces of $v$ as its *in-faces*. The left-to-right ordering of the out-edges (in-edges) of $H$ uniquely determines a left-to-right ordering of the out-faces (in-faces) of $H$. A hierarchical plane st-graph can be encoded and traversed by means of a slightly augmented DCEL: for each vertex, one simply has to add its $y$-coordinate and a pointer to its left-most out-edge (except for the sink) and a pointer to its right-most in-edge (except for the source). This augmented DCEL is easily preprocessed in linear time to obtain the peak and valley of each face. Hence, we may assume that each face is equipped with these values. Moreover, we may assume (again via preprocessing in linear time) that the vertices of the outer face are marked, so that it is possible in $\mathcal{O}(1)$ time to test whether a given vertex is external or internal. All hierarchical plane st-graphs in this paper are assumed to be represented by means of this data structure.

We say that an angle is *convex* if it is at most $\pi$ and *reflex* if it exceeds $\pi$. In a *convex* polygon, each internal angle is convex. Recall that a planar straight-line drawing of a graph is called *convex* if the boundary of each face is realized as a simple convex polygon. A *side* of a simple convex polygon is a maximal straight-line segment in its boundary, i.e., a maximal sequence of collinear edges. It is well known that a planar graph admits a convex drawing if and only if it is a subdivision of an *internally* 3*-connected* graph [31, 30, 20, 19]. There are multiple well-known equivalent definitions of this property. Each of them provides a different perspective on the concept and it will be convenient to refer to all of them. Hence, we define internal 3-connectivity in form of a characterization; a proof of the equivalence of the three properties can be found in [22].

▶ **Definition 4.** *Let $G$ be a plane 2-connected graph and let $f_o$ denote its outer face. Then $G$ is called* internally 3-connected *if and only if the following equivalent statements are satisfied:*

**(I1)** *Inserting a new vertex $v$ in $f_o$ and adding edges between $v$ and all vertices of $f_o$ results in a 3-connected graph.*

**(I2)** *From each internal vertex $w$ of $G$ there exist three paths to $f_o$ that are pairwise disjoint except for the common vertex $w$.*

**(I3)** *Every separation pair $u, v$ of $G$ is* external*, meaning that $u$ and $v$ lie on $f_o$ and every connected component of the subgraph of $G$ induced by $\mathrm{V}(G) \setminus \{u, v\}$ contains a vertex of $f_o$.*

Let $G$ be a plane graph and $\Gamma^o$ be a convex drawing of the boundary of the outer face of $G$. Even if $G$ is internally 3-connected, the drawing $\Gamma^o$ cannot necessarily be extended to a convex drawing of $G$. Recall that a path $P$ of $G$ is *arched* by a face $f$ of $G$ if there exist two distinct vertices $a, b \in \mathrm{V}(P) \cap \mathrm{V}(\partial f)$ such that the subpath of $P$ between $a$ and $b$ is not a subpath of $\partial f$, see Figure 2c. Moreover, $P$ is called *archfree* if it is not arched by an internal face of $G$. The drawing $\Gamma^o$ can be extended to a convex drawing of $G$ if and only if $G$ is a subdivision of an internally 3-connected graph and each side of $\Gamma^o$ corresponds to an archfree path of $G$ [31, 30, 20, 19]. Hong and Nagamochi [20] showed that this characterization carries over to hierarchical graphs: let $H$ be a hierarchical plane st-graph and $\Lambda^o$ be a convex drawing of the restriction of $H$ to the boundary of its outer face. If each side of $\Lambda^o$ corresponds to an archfree path of $H$, we call $\Lambda^o$ *compatible* with $H$. The drawing $\Lambda^o$ can be extended to a convex drawing of $H$ if and only if $H$ is a subdivision of an internally 3-connected graph and $\Lambda^o$ is compatible with $H$ [20]. Hong and Nagamochi also proved the following lemma.

▶ **Lemma 5** ([20, Lemma 1]). *Let $G$ be an internally 3-connected plane graph and let $f$ be an internal face of $G$. Any subpath $P$ of $\partial f$ with $|\mathrm{E}(P)| \leq |\mathrm{E}(\partial f)| - 2$ is archfree.*

## 3 Computing archfree paths efficiently

In this section, we prove Theorem 2, that is, we describe our algorithm for computing archfree paths in internally 3-connected hierarchical plane st-graphs. In fact, we prove the following generalization of Theorem 2, which gives us the freedom to influence the choice of the first edge of the computed path. This aspect will be useful in our algorithm to create convex drawings of hierarchical plane st-graphs.

▶ **Theorem 6.** *There exists an algorithm that, given an internally 3-connected hierarchical plane st-graph $G$ and (a pointer to) an internal vertex $y$ of $G$, computes an archfree directed path $P_{yz}$ from $y$ to some outer vertex $z$ of $G$ such that all vertices of $P_{yz}$ except for $z$ are internal vertices of $G$ in time $\mathcal{O}(\sum_{i \in \mathrm{V}(P_{yz}) \setminus \{z\}} \deg^+_G(i))$.*

*Moreover, the choice of the first edge $(y, u)$ of $P_{yz}$ may be influenced as follows: let $F_y$ be the set of out-faces of $y$, let $k_y = \max_{g \in F_y}\{\mathrm{peak}(g)\}$, and let $K_y = \{g \in F \mid \mathrm{peak}(g) = k\}$. We may choose $(y, u)$ to be the left-most out-edge of $v$ that is incident to the right-most out-face in $K_y$; or the right-most out-edge of $v$ that is incident to the left-most out-face in $K_y$.*

**Proof.** In the following, we describe the idea of our algorithm; for a full pseudocode version, see Algorithm 1. Suppose we have already computed a directed path $P$ from $y$ to some vertex $v$ such that all vertices of $P$ are internal (initially $v = y$). We will extend $P$ by appending an out-edge $e'$ of $v$ that is incident to an out-face of $v$ with maximum peak, for an illustration see Figure 3. More precisely, let $F$ be the set of out-faces of $v$, let $k = \max_{g \in F}\{\mathrm{peak}(g)\}$, and, finally, let $K = \{g \in F \mid \mathrm{peak}(g) = k\}$.

**(R1)** We extend $P$ by appending an out-edge $e'$ of $v$ that is incident to a face $f' \in K$.

At each edge of our path, we store a pointer to the face that was the reason why the edge was chosen, i.e., we *associate* the edge $e'$ with the face $f'$.

In general, the choice of $e'$ and $f'$ according to Rule (R1) is not unique (see Figure 3), and computing a path while exclusively relying on Rule (R1) does not necessarily lead to an archfree result. We introduce two tiebreaking rules that specialize Rule (R1). Assume for now that $v \neq y$ and let $e$ be the edge of $P$ whose head is $v$. Let $f$ be the face that is associated with $e$.

**Algorithm 1** The procedure corresponding to Theorem 6.

```
input  : an internally 3-connected hierarchical plane st-graph $G$
         an internal vertex $y$ of $G$
         $d \in \{L, R\}$ ▷ initial associated direction
output : a directed archfree path $P_{yz}$ from $y$ to an outer vertex $z$ of $G$ such that all
         vertices of $P_{yz}$ except $z$ are internal vertices of $G$

$v \longleftarrow y$ ▷ The endpoint of our current path $P$,
$P \longleftarrow \emptyset$ ▷ whose list of edges is initially empty.
$e \longleftarrow$ nil ▷ The edge that was most recently appended to $P$
$f \longleftarrow$ nil ▷ and the face associated with it.
$D \longleftarrow d$ ▷ The direction currently associated with $P$

while true do
    ▷ Let $F$ be the set of out-faces of $v$
    $k \longleftarrow \max_{g \in F}\{\text{peak}(g)\}$
    $K \longleftarrow \{g \in F \mid \text{peak}(g) = k\}$
    if $f \neq$ nil and additionally peak($f$) $= k$ then
        $e \longleftarrow$ the unique edge of $\partial f$ with tail $v$
    else ▷ $f =$ nil; or $f \neq$ nil and additionally peak($f$) $< k$
        if $D = L$ then
            $f \longleftarrow$ the right-most face in $K$
            $e \longleftarrow$ the left-most out-edge of $v$ that belongs to $\partial f$
        else ▷ $D = R$
            $f \longleftarrow$ the left-most face in $K$
            $e \longleftarrow$ the right-most out-edge of $v$ that belongs to $\partial f$
    append $e$ to $P$
    $v \longleftarrow$ head($e$)
    if $f$ is to the left of $e$ then
        $D \longleftarrow L$
    else ▷ $f$ is to the right of $e$
        $D \longleftarrow R$
    if $v$ belongs to the outer face of $G$ then
        return $P$
```

**(R2)** If $v \neq y$ and $f \in K$, we choose $f' = f$ and $e'$ to be the unique out-edge of $v$ that is incident to $f$.

In other words, we continue to follow the boundary of a face $f$ until we encounter a vertex $v$ incident to a face with strictly larger peak, see Figure 3a.

It remains to discuss the case where the preconditions of Rule (R2) are not satisfied. We *associate* a direction $D \in \{L, R\}$ with $P$, namely, $D = L$ if $f$ is to the left of $e$, and $D = R$ otherwise. Whenever we switch from $f$ to a new face, we also try to switch the direction associated with our path if possible. That is, if $D = L$, we try to choose $f'$ and $e'$ such that $f'$ is to the right of $e'$, see Figure 3b. Note that this is impossible if and only if $|K| = 1$ and the unique face $f' \in K$ is the left-face of $v$, see Figure 3c. Symmetrically, if $D = R$, we try to choose $f'$ and $e'$ such that $f'$ is to the left of $e'$, which is impossible if and only

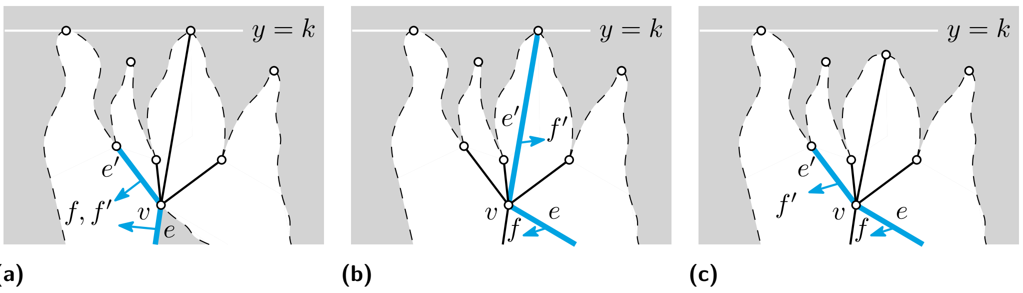


**Figure 3** The choices of $(e', f')$ made according to Rule (R1) are sometimes unique (c), but in general they are not (a–b). (a) Unique choice of $(e', f')$ according to Rule (R2). (b–c) Unique choice of $(e', f')$ according to Rule (R3). In (c) it is not possible to switch the direction of the path.

if $|K| = 1$ and the unique $f' \in K$ is the right-face of $v$. In general, this choice of $f'$ and $e'$ is still not unique. Whenever there are multiple options, we choose $f'$ and $e'$ such that if $D = L$ ($D = R$), the face $f'$ is the right-most (left-most) out-face of $v$ such that the above properties are satisfied, see Figure 3b. To streamline the algorithm, the initial path, where $v = y$ (and, hence, $f$ is undefined), is also associated with a direction $L$ or $R$. This initial direction may be freely chosen. The following rule realizes the strategy discussed in this paragraph:

**(R3)** If $v = y$ or $f \notin K$, our choice of $f'$ and $e'$ depends on $D$: if $D = L$ ($D = R$), we choose $f'$ to be the right-most (left-most) face in $K$ and $e'$ to be the left-most (right-most) out-edge of $v$ that belongs to $\partial f'$.

Indeed, whenever we switch to a new face $f' \neq f$, Rule (R3) ensures that the direction associated with the path is switched if possible:

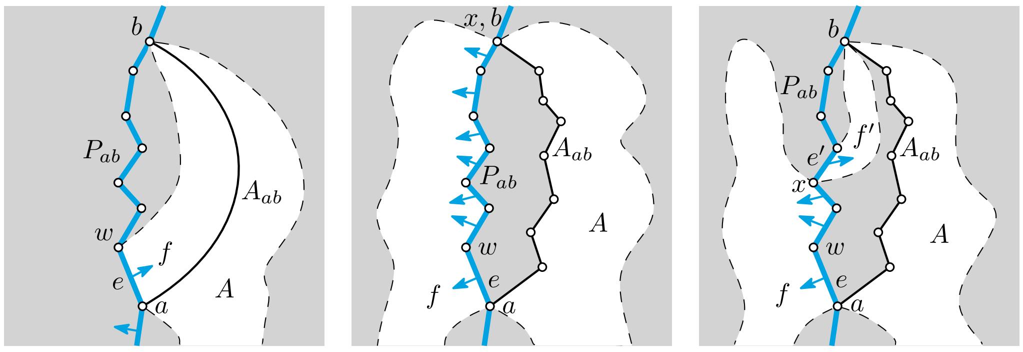
▷ Claim 7. If $D = L$ ($D = R$) and $f'$ and $e'$ are chosen according to Rule (R3), then $f'$ is to the right (left) of $e'$, unless $|K| = 1$ and the unique $f' \in K$ is the left-face (right-face) of $v$.

**Correctness.** Since $G$ has the st-property, Algorithm 1 terminates with a directed path $P_{yz}$ from $y$ to some outer vertex $z$ of $G$ such that all vertices of $P$ except for $z$ are internal vertices of $G$. Since the initial direction may be freely chosen, Rule (R3) guarantees the choice of the first edge of $P_{yz}$ as described in the statement of the theorem.

It remains to show that $P_{yz}$ is archfree. To the contrary, assume that $P_{yz}$ is arched by a face $A$. Without loss of generality, we may assume that $A$ is to the right of $P_{yz}$. This implies the existence of a directed subpath $A_{ab}$ of the left boundary of $A$ that starts at a vertex $a \in \mathrm{V}(P_{yz})$, ends at a vertex $b \in \mathrm{V}(P_{yz})$ with $b \neq a$, and is interior-disjoint from $P_{yz}$. Let $P_{ab}$ denote the subpath of $P_{yz}$ that leads from $a$ to $b$. Let $C_{ab}$ denote the simple cycle formed by $P_{ab}$ and $A_{ab}$ in the underlying undirected graph of $G$.

In a plane 2-connected graph, every edge belongs to the boundary of at least one internal face. Consequently, Lemma 5 implies that in internally 3-connected graphs every path of length 1 is archfree (since each face is bounded by at least three edges). Since $G$ is internally 3-connected, it follows that $|\mathrm{E}(P_{ab})| \geq 2$. Let $e = (a, w)$ denote the unique out-edge of $a$ that belongs to $P_{ab}$ (where $w \neq b$ since $|P_{ab}| \geq 2$). Let $f$ denote the face associated with $e$. By the definition of $A_{ab}$, we have $f \neq A$. The face $f$ can be either to the left or the right of $e$. Accordingly, we distinguish two cases.

**Case 1:** $f$ is to the right of $e$. For an illustration see Figure 4a. This implies that $f$ is interior to the cycle $C_{ab}$. Consequently, we have $\mathrm{peak}(f) \leq \mathrm{peak}(A)$. Moreover, by Rule (R1), we have $\mathrm{peak}(f) = \mathrm{peak}(A)$ since both $A$ and $f$ are out-faces of $a$. From the fact that $f$ is interior to the cycle $C_{ab}$, we can also conclude that $a$ is the (unique) source of $f$. Therefore, the face $f$ and the edge $e$ were chosen according to Rule (R3) (rather than Rule (R2)). Due to the fact that $\mathrm{peak}(f) = \mathrm{peak}(A)$, Claim 7 implies that the direction associated with the subpath $P_{ya}$ of $P_{yz}$ from $y$ to $a$ is $L$. Consequently, Rule (R3) implies that $f$ is the right-most out-face of $v$ whose peak is $\mathrm{peak}(f)$ ($= \mathrm{peak}(A)$). We obtain a contradiction to the fact that $A$ is to the right of $f$. ◁



**(a)** **(b)** **(c)**

**Figure 4** (a) Case 1 and (b–c) Case 2 in the proof of Theorem 6.

**Case 2:** $f$ is to the left of $e$. Let $P_{ax}$ denote the unique maximal subpath of $P_{ab}$ such that all edges of $P_{ax}$ are associated with $f$.

To the contrary, assume that the endpoint $x \neq a$ of $P_{ax}$ is $b$; for an illustration see Figure 4b. It follows that $a$ and $b$ form a separator in $G$ that separates $w$ (and all other vertices in the closed interior of $C_{ab}$ except for $a$ and $b$) from the outer vertices of $G$. Consequently, $a$ and $b$ form a nonexternal separation pair; a contradiction to the internal 3-connectivity of $G$. Therefore, we have $x \neq b$.

Let $e'$ denote the unique out-edge of $x$ that belongs to $P_{ab}$; for an illustration see Figure 4c. By Rule (R2), the edge $e'$ is associated with a face $f'$ such that $\mathrm{peak}(f') > \mathrm{peak}(f)$. Therefore, face $f'$ and edge $e'$ were chosen according to Rule (R3). Since $f$ is to the left of $e$, each edge of $P_{ax}$ has $f$ to its left. Therefore the direction associated with the subpath $P_{yx}$ of $P_{yz}$ from $y$ to $x$ is $L$. By Rule (R1) applied to $a$, we have $\mathrm{peak}(f) \geq \mathrm{peak}(A)$. Moreover, we have $\mathrm{peak}(A) > \mathrm{y}(x)$ since $x \neq b$. Consequently, $\mathrm{peak}(f) > \mathrm{y}(x)$, which implies that $f$ is the left-face of $x$. Therefore, by Claim 7, the face $f'$ is to the right of $e'$. This implies that $f'$ is interior to $C_{ab}$ and, consequently, $\mathrm{peak}(f') \leq \mathrm{peak}(A)$. Altogether, we have $\mathrm{peak}(f) < \mathrm{peak}(f') \leq \mathrm{peak}(A) \leq \mathrm{peak}(f)$; a contradiction. ◁

**Running time.** The claimed running time of $\mathcal{O}(\sum_{i \in \mathrm{V}(P_{yz}) \setminus \{z\}} \deg^+_G(i))$ of Algorithm 1 is easy to achieve: the initialization takes $\mathcal{O}(1)$ time. The while loop is executed once for each vertex of $P_{yz}$ that is an internal vertex of $G$. For each vertex $v$, the quantity $k$ can be computed in time $\mathcal{O}(\deg^+_G(v))$ by sweeping the linear sequence of out-faces of $v$ once. With a second sweep of the sequence, we can then determine in time $\mathcal{O}(\deg^+_G(v))$ the set $K$. Actually, it suffices to remember the left-most and the right-most out-face in $K$. With this information, the remaining lines in the while loop can be executed in time $\mathcal{O}(1)$. ◀

## 4 Computing convex drawings of hierarchical graphs in linear time

In this section, we describe our $\mathcal{O}(n)$-time algorithm to create a convex drawing of a hierarchical plane st-graph. We follow the idea of the recursive combinatorial construction by Hong and Nagamochi [20]. As discussed in Section 1, the main improvement comes from using Theorem 6 to compute archfree paths. We also need to make some changes to the low-level details of the original algorithm and apply a more careful (amortized) analysis of its running time. The correctness of our algorithm follows for the most part from the correctness of the algorithm by Hong and Nagamochi.

▶ **Theorem 1.** *There exists an algorithm that, given a subdivision $G = (V, E)$ of an internally 3-connected hierarchical plane st-graph and a convex polygon $\Gamma^o$ that is compatible with $G$, computes a convex drawing of $G$ with $\Gamma^o$ as the realization of the outer face in time $\mathcal{O}(n)$ where $n = |V|$.*

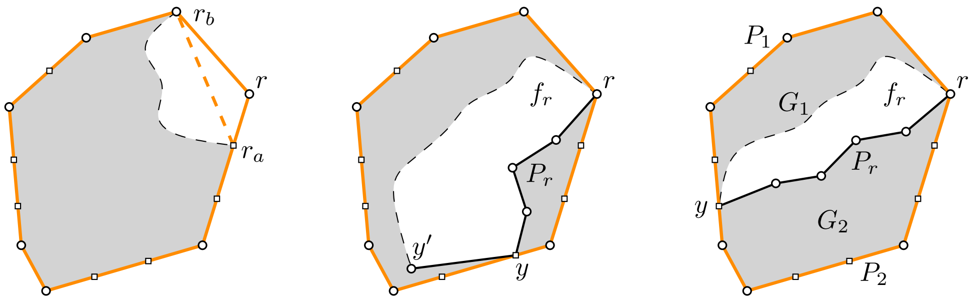
**Proof sketch.** It suffices to prove the claim for the case that $G$ is internally 3-connected. The $y$-coordinate of each vertex in the desired drawing of $G$ is already fixed. Moreover, the $x$-coordinates of the vertices of the outer face are also fixed by $\Gamma^o$. Hence, our goal is to (recursively) compute the $x$-coordinates of the internal vertices in a convex drawing of $G$ with $\Gamma^o$ as the realization of the outer face.

We assume we are given a cyclic list $L$ that contains the vertices of $G$ corresponding to the vertices of $\Gamma^o$ whose outer angles are reflex in the order in which they appear around $\Gamma^o$. We refer to $L$ as the *corner list* of $\Gamma^o$. Initially, we can preprocess $G$ and $\Gamma^o$ to obtain $L$ in $\mathcal{O}(n)$ time. When making recursive calls, we will split $L$ appropriately to create a new corner list for each subproblem in $\mathcal{O}(1)$ time. Given the list $L$, it is possible in $\mathcal{O}(1)$ time to compute a vertex $r$ that corresponds to a vertex of $\Gamma^o$ whose outer angle is reflex and such that $r$ is not the source or sink of $G$. Without loss of generality, we assume that $r$ belongs to the right boundary of $\Gamma^o$. We distinguish two cases regarding the degree of $r$.

**Case 1:** $\deg_G(r) = 2$. For an illustration see Figure 5a. Let $r_a$ and $r_b$ denote the in-neighbor and out-neighbor of $r$, respectively. If $|L| = 3$ and $r_a$ is the source of $G$ and $r_b$ is the sink of $G$, then $G$ has no internal vertices (otherwise, the internal face incident to $r$ would arch the path corresponding to the side $r_a r_b$ of $\Gamma^o$ thereby contradicting the compatibility of $\Gamma^o$). Hence, we simply report that the coordinates of all internal vertices are already fixed. This is the base case of our recursion. It can be recognized and dealt with in $\mathcal{O}(1)$ time.

So assume that $|L| \geq 4$ or that $r$ has a neighbor that is not the sink or source of $G$. Let $\Gamma_1^o$ denote the simple (convex) polygon obtained from $\Gamma^o$ created by replacing the segments $rr_a$ and $rr_b$ with the segment $r_a r_b$. The simplicity of $\Gamma_1^o$ is implied by the above assumption. If $(r_a, r_b) \in E$ we set $G_1' = G$. Otherwise, we add the edge $(r_a, r_b)$ in the internal face incident to $r$ and call the resulting graph $G_1'$. We delete $r$ from $G_1'$ and call the resulting graph $G_1$. Properties (I1)–(I2) of Definition 4 can be used to show that $G_1$ is internally 3-connected. Moreover, it can be derived by Lemma 5 that $\Gamma_1^o$ is compatible with $G_1$. We recursively compute the coordinates of the internal vertices in a convex drawing of $G_1$ with $\Gamma_1^o$ as the realization of the outer face. These coordinates combined with the coordinates of $r$ correspond to the desired drawing of $G$. The graph $G$ and its polygon $\Gamma^o$ are easily transformed into $G_1$ and $\Gamma_1^o$ in $\mathcal{O}(1)$ time, and $L$ can be transformed into a corner list of $\Gamma_1^o$ in $\mathcal{O}(1)$ time. ◁

**Case 2:** $\deg_G(r) \geq 3$. For illustrations see Figures 5b, 5c, and 6a Without loss of generality, there is an edge whose head is $r$ and whose tail is an internal vertex of $G$. Let $f_r$ denote the left-face of $r$ and let $P_r'$ denote the (unique) directed subpath of $\partial f_r$ that connects the source $y'$ of $f_r$ with $r$. Let $y$ be the first outer vertex of $G$ distinct from $r$ that is encountered when traversing $P'_r$ from $r$ towards $y'$ – if no such vertex exists, we set $y = y'$. We define $P_r$ to be the subpath of $P'_r$ between $y$ and $r$. If $y$ and $r$ are nonadjacent, we have $|\mathrm{E}(P_r)| \leq |\mathrm{E}(\partial f_r)| - 2$ and hence $P_r$ is archfree by Lemma 5. If $y$ and $r$ are adjacent, then $P_r = (y, r)$; otherwise, by the definition of $f_r$, the edge $(y, r)$ is to the right of $P_r$, which implies that the vertices $r, y$ form a nonexternal separation pair that separates the internal vertices of $P_r$ from the outer face and, thus, contradicts the internal 3-connectivity of $G$. Hence, $P_r$ is again archfree by Lemma 5. So in any case $P_r$ is archfree. Moreover, $P_r$ is easily computed in $\mathcal{O}(|\mathrm{E}(P_r)|)$ time.



**(a)** Case 1. **(b)** Case 2.1. **(c)** Case 2.1.

**Figure 5** Case 1 and Case 2.1 in the proof of Theorem 1 and [20, Theorem 8]. (b) depicts the case where $y \neq y'$ and $P_r \neq P'_r$, whereas (c) depicts the case where $y = y'$ and $P_r = P'_r$.

We distinguish two cases depending on whether $y$ is an internal vertex or not.

**Case 2.1:** $y$ is an outer vertex of $G$. For illustrations see Figures 5b and 5c. The boundary of the outer face of $G$ contains two interior disjoint paths $P_1, P_2$ between $r$ and $y$. Each of these two paths forms a cycle together with $P_r$. The closed interior of each of these two cycles describes a hierarchical plane st-graph. We denote these two graphs by $G_1$ and $G_2$ such that $G_i$, $i \in \{1, 2\}$ has $P_i$ on its outer face. We define $\Gamma^o_1$ to be the polygon resulting from replacing the part of $\Gamma^o$ that corresponds to $P_2$ with $P_r$ drawn as a straight-line segment, thereby fixing the coordinates of the internal vertices of $P_r$. The drawing $\Gamma^o_2$ is defined analogously. Since each side of $\Gamma^o$ corresponds to an archfree path in $G$, it follows that $r$ and $y$ do not belong to a common side of $\Gamma^o$. Hence, $\Gamma^o_1$ and $\Gamma^o_2$ are simple (convex) polygons. Moreover, since $P_r$ is archfree, $\Gamma^o_1$ is compatible with $G_1$ and $\Gamma^o_2$ is compatible with $G_2$. By Property (I2) of Definition 4 for $G$ it easily follows that $G_1$ and $G_2$ are internally 3-connected. We recursively compute the coordinates of the internal vertices in convex drawings of $G_1$ and $G_2$ with outer face $\Gamma_1$ and $\Gamma_2$, respectively. Note that these coordinates combined with the coordinates of $\Gamma^o$ and $P_r$ correspond to the desired drawing of $G$.

The hierarchical plane st-graphs $G_1$ and $G_2$ can be obtained in $\mathcal{O}(|\mathrm{E}(P_r)|)$ time by splitting $G$ along $P_r$. The polygons $\Gamma^o_1$ and $\Gamma^o_2$ can be obtained in $\mathcal{O}(|\mathrm{E}(P_r)|)$ time by splitting $\Gamma^o$ and their corner lists can be obtained in $\mathcal{O}(1)$ time by splitting $L$. ◁

**Case 2.2:** $y$ is an internal vertex of $G$. For an illustration see Figure 6a. We compute a directed archfree path $P_x$ from a vertex $x$ on the outer face of $G$ to $y$ by using (a symmetric version of) Theorem 2. The paths $P_x$ and $P_r$ are disjoint except for the common endpoint $y$. We also compute a directed archfree path $P_z$ from $y$ to a vertex $z$ on the outer face of $G$ by using Theorem 6. To ensure that $P_z$ is disjoint from $P_r$ (except for the common endpoint $y$), we make use of the ability to influence the choice of the first edge $(y, u)$ of $P_z$ as guaranteed by Theorem 6. Let $F_y$ be the set of out-faces of $y$, let $k_y = \max_{g \in F_y}\{\mathrm{peak}(g)\}$, and let $K_y = \{g \in F \mid \mathrm{peak}(g) = k\}$. If $K_y = \{f_r\}$, we pick the left-most out-edge of $y$ that is incident to the right-most out-face in $K_y$ (which is $f_r$). Since $f_r$ is the left-face of $r$, this choice guarantees that $P_z$ and $P_r$ are disjoint. Otherwise (if $K_y$ contains at least one face distinct from $f_r$), we pick the right-most out-edge of $y$ that is incident to the left-most out-face in $K_y$. Since $f_r$ is the left-face of $r$, its peak is strictly larger than $\mathrm{y}(r)$. Hence, $K_y$ cannot contain any face that is to the right of $f_r$. Consequently, it contains a face to the left of $f_r$. Thus, our choice guarantees that $P_z$ is disjoint from $P_r$ since $P_r$ is the left-face of $r$.

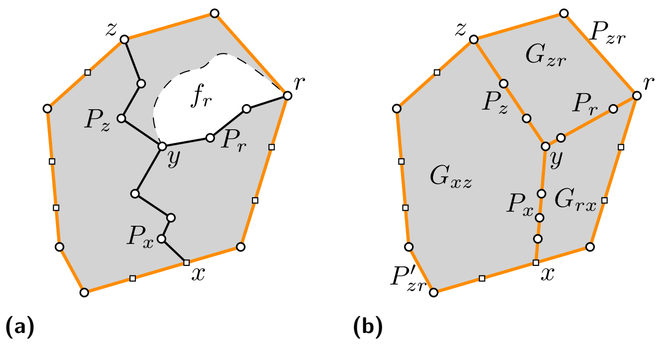


**Figure 6** Case 2.2 in the proof of Theorem 1 and [20, Theorem 8].

The boundary of the outer face of $G$ contains two interior disjoint paths from $z$ to $r$. Let $P_{zr}$ denote the path that does not contain $x$ and let $P'_{zr}$ denote the path that contains $x$, for an illustration see Figure 6b. The closed interior of the cycle formed by $P_{zr}, P_z$, and $P_r$ describes a hierarchical plane st-graph $G_{zr}$, which is easily seen to be internally 3-connected due to Property (I2) of Definition 4. We pick a point $p$ with $\mathrm{y}(p) = \mathrm{y}(y)$ in the interior of the triangle formed by the vertices $r, x, z$. Finally, we create a convex polygon $\Gamma^o_{zr}$ from $\Gamma^o$ by replacing the part corresponding to $P'_{zr}$ with the straight-line segments $pz$ and $pr$, which fixes the coordinates of the vertices of $P_r$ and $P_z$. Since $P_r$ and $P_z$ are archfree in $G$, it follows that $\Gamma^o_{zr}$ is compatible with $G_{zr}$. Analogously, we define two other internally 3-connected hierarchical plane st-graphs $G_{rx}$ and $G_{xz}$ that together with $G_{zr}$ partition the internal faces of $G$. We also construct two convex polygons $\Gamma^o_{rx}$ and $\Gamma^o_{xz}$ that are compatible with $G_{rx}$ and $G_{xz}$, respectively, that also use the point $p$ as a corner. We recursively compute the internal coordinates of convex drawings of $G_{zr}, G_{rx}$, and $G_{xz}$ with $\Gamma^o_{zr}$, $\Gamma^r_{rx}$ and $\Gamma^o_{xz}$ as the realization of the outer face, respectively. Note that these coordinates combined with the coordinates of $\Gamma^o$, $P_r$, $P_z$, and $P_x$ correspond to the desired drawing of $G$.

By Theorem 6, the path $P_x$ can be computed in $\mathcal{O}(\sum_{i\in \mathrm{V}(P_x)\setminus\{x\}} \deg^-_G(i))$ time and $P_z$ can be computed in $\mathcal{O}(\sum_{i\in \mathrm{V}(P_z)\setminus\{z\}} \deg^+_G(i))$ time. By splitting $G$ and $\Gamma^o$, we can compute $G_{zr}, G_{rx}$, and $G_{xz}$ and $\Gamma^o_{zr}$, $\Gamma^r_{rx}$, and $\Gamma^o_{xz}$, respectively, in $\mathcal{O}(|\mathrm{E}(P_r)| + |\mathrm{E}(P_x)| + |\mathrm{E}(P_z)|)$ time. By splitting $L$, we can compute the corner lists of $\Gamma^o_{zr}$, $\Gamma^r_{rx}$, and $\Gamma^o_{xz}$ in $\mathcal{O}(1)$ time. $\lhd$

**Running time.** The preprocessing (computing $L$) takes $\mathcal{O}(n)$ time. Case 1 can be dealt with in $\mathcal{O}(1)$ time. The time to take care of Case 2.1 can be expressed as $\mathcal{O}(1)+\mathcal{O}(|\mathrm{E}(P_r)|)$. Case 2.2 can be taken care of in time $\mathcal{O}(1)+\mathcal{O}(|\mathrm{E}(P_r)|+\sum_{i\in \mathrm{V}(P_x)\setminus\{x\}} \deg^-_G(i)+\sum_{i\in \mathrm{V}(P_z)\setminus\{z\}} \deg^+_G(i))$. In all three cases, the nonconstant summands only involve internal vertices of $G$, that is, $|\mathrm{E}(P_r)|$ is linear in the number of vertices of $P_r$ that are internal vertices of $G$, and the expressions $\sum_{i\in \mathrm{V}(P_x)\setminus\{x\}} \deg^-_G(i)$ and $\sum_{i\in \mathrm{V}(P_z)\setminus\{z\}} \deg^+_G(i))$ only involve the degree of vertices of $P_x$ and $P_z$, respectively, that are internal to $G$. The graph $G$ is then split to obtain up to three hierarchical plane st-graphs in which these internal vertices are external. Hence, each vertex of $G$ can contribute to the nonconstant part of the running time of at most one subproblem in the recursion tree. Therefore, the total sum of these nonconstant parts is bounded by $\sum_{v\in V} \deg_G(v) \subseteq \mathcal{O}(n)$. It remains to find an upper bound on the total number of subproblems. The number of subproblems for which Case 1 arises is bounded by the number of faces of an internal triangulation of $G$, which is $\mathcal{O}(n)$. Each subproblem for which Case 2 arises splits at least one internal edge, which becomes external in the created subproblems. Hence, each edge is split at most once and so the number of the subproblems for which Case 2 arises is $\mathcal{O}(n)$. Therefore the total running time is $\mathcal{O}(n)$. ◀

## References

1 Oswin Aichholzer, Greg Aloupis, Erik D. Demaine, Martin L. Demaine, Vida Dujmovic, Ferran Hurtado, Anna Lubiw, Günter Rote, André Schulz, Diane L. Souvaine, and Andrew Winslow. Convexifying polygons without losing visibilities. In Greg Aloupis and Bremner, editors, *Canadian Conference on Computational Geometry (CCCG)*, pages 229–234, 2011.

2 Soroush Alamdari, Patrizio Angelini, Fidel Barrera-Cruz, Timothy M. Chan, Giordano Da Lozzo, Giuseppe Di Battista, Fabrizio Frati, Penny Haxell, Anna Lubiw, Maurizio Patrignani, Vincenzo Roselli, Sahil Singla, and Bryan T. Wilkinson. How to morph planar graph drawings. *SIAM J. Comput.*, 46(2):824–852, 2017. `doi:10.1137/16M1069171`.

3 Josh Alman and Virginia Vassilevska Williams. A refined laser method and faster matrix multiplication. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 522–539. SIAM, 2021. `doi:10.1137/1.9781611976465.32`.

4 Noga Alon and Raphael Yuster. Matrix sparsification and nested dissection over arbitrary fields. *Journal of the ACM*, 60(4):25, 2013.

5 Patrizio Angelini, Giordano Da Lozzo, Giuseppe Di Battista, Fabrizio Frati, and Vincenzo Roselli. The importance of being proper: (in clustered-level planarity and t-level planarity). *Theor. Comput. Sci.*, 571:1–9, 2015. `doi:10.1016/j.tcs.2014.12.019`.

6 Patrizio Angelini, Giordano Da Lozzo, Giuseppe Di Battista, Fabrizio Frati, Maurizio Patrignani, and Vincenzo Roselli. Morphing planar graph drawings optimally. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, volume 8572 of *Lecture Notes in Computer Science*, pages 126–137. Springer, 2014. `doi:10.1007/978-3-662-43948-7_11`.

7 Patrizio Angelini, Giordano Da Lozzo, Fabrizio Frati, Anna Lubiw, Maurizio Patrignani, and Vincenzo Roselli. Optimal morphs of convex drawings. In Lars Arge and János Pach, editors, *31st International Symposium on Computational Geometry, SoCG 2015, June 22-25, 2015, Eindhoven, The Netherlands*, volume 34 of *LIPIcs*, pages 126–140. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015. `doi:10.4230/LIPIcs.SOCG.2015.126`.

8 Guido Brückner and Ignaz Rutter. Partial and constrained level planarity. In Philip N. Klein, editor, *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 2000–2011. SIAM, 2017. `doi:10.1137/1.9781611974782.130`.

9 Erin Wolf Chambers, Jeff Erickson, Patrick Lin, and Salman Parsa. How to morph graphs on the torus. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 2759–2778. SIAM, 2021. `doi:10.1137/1.9781611976465.164`.

10 Marek Chrobak, Michael T. Goodrich, and Roberto Tamassia. Convex drawings of graphs in two and three dimensions (preliminary version). In Sue Whitesides, editor, *Proceedings of the Twelfth Annual Symposium on Computational Geometry, Philadelphia, PA, USA, May 24-26, 1996*, pages 319–328. ACM, 1996. `doi:10.1145/237218.237401`.

11 Robert Connelly, Erik D. Demaine, and Günter Rote. Straightening polygonal arcs and convexifying polygonal cycles. *Discrete & Computational Geometry*, 30:205–239, 2003.

12 Mark de Berg, Otfried Cheong, Marc J. van Kreveld, and Mark H. Overmars. *Computational geometry: algorithms and applications, 3rd Edition*. Springer, 2008. URL: https://www.worldcat.org/oclc/227584184.

13 Giuseppe Di Battista and Enrico Nardelli. Hierarchies and planarity theory. *IEEE Trans. Systems, Man, and Cybernetics*, 18(6):1035–1046, 1988. doi:10.1109/21.23105.

14 Peter Eades, Qing-Wen Feng, Xuemin Lin, and Hiroshi Nagamochi. Straight-line drawing algorithms for hierarchical graphs and clustered graphs. *Algorithmica*, 44(1):1–32, 2006. doi:10.1007/s00453-004-1144-8.

15 Michael S. Floater. Parameterization and smooth approximation of surface triangulations. *Computer Aided Geometric Design*, 14(3):231–250, 1997.

16 Michael S. Floater. Parametric tilings and scattered data approximation. *International Journal of Shape Modeling*, 4(03n04):165–182, 1998.

17 Jonas Gomes, Lucia Darsa, Bruno Costa, and Luiz Velho. *Warping and Morphing of Graphical Objects*. Morgan Kaufmann, 1999.

18 Lenwood S. Heath and Sriram V. Pemmaraju. Recognizing leveled-planar dags in linear time. In Franz-Josef Brandenburg, editor, *Graph Drawing, Symposium on Graph Drawing, GD '95, Passau, Germany, September 20-22, 1995, Proceedings*, volume 1027 of *Lecture Notes in Computer Science*, pages 300–311. Springer, 1995. doi:10.1007/BFb0021813.

19 Seok-Hee Hong and Hiroshi Nagamochi. Convex drawings of graphs with non-convex boundary constraints. *Discret. Appl. Math.*, 156(12):2368–2380, 2008. doi:10.1016/j.dam.2007.10.012.

20 Seok-Hee Hong and Hiroshi Nagamochi. Convex drawings of hierarchical planar graphs and clustered planar graphs. *J. Discrete Algorithms*, 8(3):282–295, 2010. doi:10.1016/j.jda.2009.05.003.

21 Michael Jünger, Sebastian Leipert, and Petra Mutzel. Level planarity testing in linear time. In Sue Whitesides, editor, *Graph Drawing, 6th International Symposium, GD'98, Montréal, Canada, August 1998, Proceedings*, volume 1547 of *Lecture Notes in Computer Science*, pages 224–237. Springer, 1998. doi:10.1007/3-540-37623-2_17.

22 Linda Kleist, Boris Klemz, Anna Lubiw, Lena Schlipf, Frank Staals, and Darren Strash. Convexity-increasing morphs of planar graphs. *Comput. Geom.*, 84:69–88, 2019. doi:10.1016/j.comgeo.2019.07.007.

23 Boris Klemz and Günter Rote. Ordered level planarity and its relationship to geodesic planarity, bi-monotonicity, and variations of level planarity. *ACM Trans. Algorithms*, 15(4):53:1–53:25, 2019. doi:10.1145/3359587.

24 François Le Gall. Powers of tensors and fast matrix multiplication. In *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation*, pages 296–303. ACM, 2014.

25 Richard J. Lipton, Donald J. Rose, and Robert Endre Tarjan. Generalized nested dissection. *SIAM J. Numerical Analysis*, 16(2):346–358, 1979.

26 Richard J. Lipton and Robert Endre Tarjan. Applications of a planar separator theorem. *SIAM J. Comput.*, 9(3):615–627, 1980. doi:10.1137/0209046.

27 Giordano Da Lozzo, Giuseppe Di Battista, Fabrizio Frati, Maurizio Patrignani, and Vincenzo Roselli. Upward planar morphs. *Algorithmica*, 82(10):2985–3017, 2020. doi:10.1007/s00453-020-00714-6.

28 János Pach and Géza Tóth. Monotone drawings of planar graphs. *J. Graph Theory*, 46(1):39–47, 2004. doi:10.1002/jgt.10168.

29 André Schulz. Drawing 3-polytopes with good vertex resolution. *J. Graph Algorithms Appl.*, 15(1):33–52, 2011. doi:10.7155/jgaa.00216.

30 Carsten Thomassen. Plane representations of graphs. In J. A. Bondy and U. S. R. Murty, editors, *Progress in Graph Theory*, pages 43–69. Academic Press, 1984.

31 William T. Tutte. Convex representations of graphs. *Proceedings of the London Mathematical Society*, s3-10(1):304–320, 1960. doi:10.1112/plms/s3-10.1.304.

32 William T. Tutte. How to draw a graph. *Proceedings of the London Mathematical Society*, 3(1):743–767, 1963.