Computational Geometry 43 (2010) 191–206

Contents lists available at ScienceDirect

Computational Geometry: Theory and Applications

www.elsevier.com/locate/comgeo

An algorithm for constructing star-shaped drawings of plane graphs

Seok-Hee Hong [a,*], Hiroshi Nagamochi [b]

[a] School of Information Technologies, University of Sydney, Australia
[b] Department of Applied Mathematics and Physics, Graduate School of Informatics, Kyoto University, Japan

ARTICLE INFO

Article history:
Received 6 October 2008
Accepted 30 June 2009
Available online 3 July 2009
Communicated by T. Tokuyama

Keywords:
Graph drawing
Convex drawing
Star-shaped drawing
Plane graphs
Biconnected plane graphs
Concave corner

ABSTRACT

A straight-line planar drawing of a plane graph is called a *convex drawing* if every facial cycle is drawn as a convex polygon. Convex drawings of graphs is a well-established aesthetic in graph drawing, however not all planar graphs admit a convex drawing. Tutte [W.T. Tutte, Convex representations of graphs, Proc. of London Math. Soc. 10 (3) (1960) 304–320] showed that every triconnected plane graph admits a convex drawing for any given boundary drawn as a convex polygon. Thomassen [C. Thomassen, Plane representations of graphs, in: Progress in Graph Theory, Academic Press, 1984, pp. 43–69] gave a necessary and sufficient condition for a biconnected plane graph with a prescribed convex boundary to have a convex drawing.

In this paper, we initiate a new notion of *star-shaped drawing* of a plane graph as a straight-line planar drawing such that each inner facial cycle is drawn as a star-shaped polygon, and the outer facial cycle is drawn as a convex polygon. A star-shaped drawing is a natural extension of a convex drawing, and a new aesthetic criteria for drawing planar graphs in a convex way as much as possible. We give a sufficient condition for a given set A of corners of a plane graph to admit a star-shaped drawing whose concave corners are given by the corners in A, and present a linear time algorithm for constructing such a star-shaped drawing.

1. Introduction

Graph drawing has attracted much attention over the last twenty years due to its wide range of applications, such as VLSI design, social networks, software engineering and bioinformatics. Two or three dimensional drawings of graphs with a variety of aesthetics and edge representations have been extensively studied [5,17,20,21].

In this paper, we only consider a drawing drawn in two dimensions. A graph G is called *planar* if it has an embedding in the plane without edge crossings, which is called a *plane embedding* of G. A plane embedding of a planar graph is called *a straight-line drawing* if all the edges of the graph are drawn as straight-line segments. Straight-line drawing is one of the most popular drawing conventions in graph drawing. It is known that every planar graph admits a straight-line drawing [7,22,26]. There are algorithms for constructing straight-line drawings of planar graphs with various drawing aesthetics (e.g., [5,17,20]).

A straight-line drawing is called a *convex drawing* if every facial cycle is drawn as a convex polygon. Convex representation of graphs is a well-established aesthetic in graph drawing, but not all planar graphs admit a convex drawing. Tutte [24] showed that every triconnected plane graph admits a convex drawing for any given boundary drawn as a convex polygon, and presented a barycenter mapping method to construct such a convex drawing. Thomassen [23] gave a necessary and

* Corresponding author.
E-mail addresses: shhong@it.usyd.edu.au (S.-H. Hong), nag@amp.i.kyoto-u.ac.jp (H. Nagamochi).

0925-7721/$ – see front matter
doi:10.1016/j.comgeo.2009.06.008

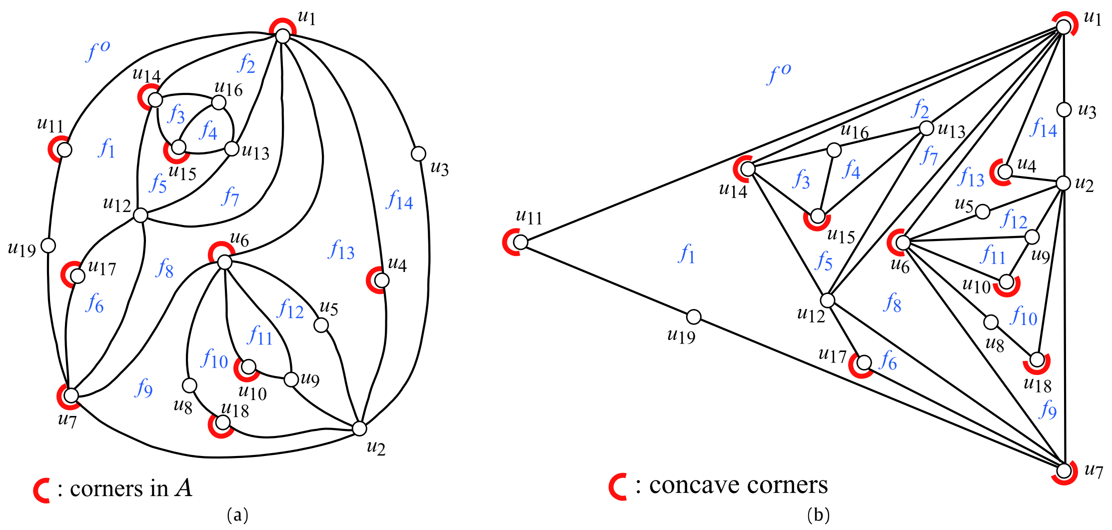

Fig. 1. (a) A biconnected plane graph G; (b) A star-shaped drawing D of G.

sufficient condition for a biconnected plane graph with a prescribed convex boundary to have a convex drawing. Based on this result, Chiba et al. [2] presented a linear-time algorithm for finding a convex drawing (if any) for a biconnected plane graph with a specified convex boundary.

In general, the convex drawing problem has been well investigated, such as the problem of constructing *convex grid drawings* of graphs [1,3,4,18]. Every triconnected plane graph has a convex grid drawing on an $(n-2)\times(n-2)$ grid, and such a grid drawing can be found in linear time [4]. A linear-time algorithm for finding a convex grid drawing of four-connected plane graphs with four or more vertices on the outer face was presented by Miura et al. [18]. Miura et al. [19] gave a linear-time algorithm for finding a convex drawing with the minimum number of outer apices for an internally triconnected plane graph. Convex drawings of hierarchical planar graphs and clustered planar graphs have also been investigated [11].

However, not much attention has been paid to the problem of finding a “nearly convex drawing” with a non-convex boundary or non-convex faces. Recently, we proved that every triconnected plane graph with a fixed star-shaped polygon boundary has an *inner-convex drawing* (i.e., a drawing in which every inner face is drawn as a convex polygon) [9,10]. Note that this is an extension of the classical result by Tutte [24], since any convex polygon is a star-shaped polygon.

To draw biconnected graphs, which do not admit convex drawings, in a convex way as much as possible, it is natural to minimize the number of non-convex faces or concave corners in a drawing. However, Kant [16] already proved the NP-completeness of the problem of deciding whether a biconnected planar/plane graph can be drawn with at most k non-convex faces.

In this paper, we initiate a new notion of *star-shaped drawing* of a plane graph as a straight-line drawing such that each inner facial cycle is drawn as a star-shaped polygon, and the outer facial cycle is drawn as a convex polygon. See Fig. 1(b) for an example of a star-shaped drawing. Note that convexity is a well-established aesthetic criteria for drawing graphs, however most of biconnected planar graphs do not admit a convex drawing [23]. A star-shaped drawing is a natural extension of a convex drawing, and a new aesthetic criteria for drawing biconnected planar graphs in a convex way as much as possible.

Note that constructing any star-shaped drawing of a biconnected plane graph is easy to achieve using augmentation. First, we augment the biconnected plane graph into a triconnected plane graph using a *star triangulation* (i.e., for each non-triangular face, we add a new vertex, and add new edges between the vertex and all the vertices of the face). Then we can construct a convex drawing of the new graph using Tutte’s algorithm [24]. Finally, we remove the newly added vertices and edges from the convex drawing, resulting in a star-shaped drawing of the original graph.

In this paper, we study star-shaped drawings of graphs with a *prescribed* set of concave corners. More specifically, we give a sufficient condition for a subset A of corners of a plane graph to admit a star-shaped drawing whose concave corners are given by the corners in A. See Fig. 1(a) for an example of a plane graph and a subset A of concave corners. The condition is based on the structure of plane graphs, called *configuration* and *fringe corners*. We then present a linear-time algorithm for computing a star-shaped drawing of a given biconnected plane graph with a prescribed set of concave corners. We use a decomposition of a biconnected graph into triconnected components for the drawing algorithm.

In our companion papers [12–14], we studied several problems on finding a star-shaped drawing with the *minimum number of concave corners*. Given a biconnected *planar* graph, we considered the problem of finding the best plane embedding of the graph, i.e., the embedding that gives the minimum number of concave corners, and gave a linear-time algorithm based on a lower bound on the optimal value [12]. Given a biconnected *plane* graph G, we derived a necessary and sufficient condition for a set B of corners of G to admit a star-shaped drawing whose concave corners are *contained* in B, and gave a linear-time algorithm for finding such a star-shaped drawing [13]. Given a biconnected plane graph G with non-negative costs on corners, a star-shaped drawing that minimizes the total cost of concave corners can be found in linear time [14]. Our algorithm for constructing a star-shaped drawing of a graph with a given set of concave corners in this paper is a key ingredient to all the above linear-time algorithms [12–14], which compute a good set A of concave corners.

This paper is organized as follows. In Section 2, we review basic terminology on plane graphs, star-shaped drawings and SPQR trees. In Section 3, we formally define "configuration" and "fringe corners." In Section 4, we prove the main result of this paper by presenting an algorithm to construct a star-shaped drawing with a prescribed set of corners which satisfies a sufficient condition. In Section 5, we make concluding remarks.

2. Preliminaries

Throughout the paper, a graph $G = (V, E)$ is a simple undirected graph. The set of vertices and the set of edges of a graph G are denoted by $V(G)$ and $E(G)$, respectively. The set of edges incident to a vertex $v \in V$ is denoted by $E(v)$. The degree of a vertex v in G is denoted by $d_G(v)$ (i.e., $d_G(v) = |E(v)|$). For a subset $X \subseteq E$ (respectively, $X \subseteq V$), let $G - X$ denote the graph obtained from G by removing the edges in X (respectively, the vertices in X together with the edges in $\bigcup_{v\in X} E(v)$).

2.1. Plane graphs and biconnected plane graphs

A graph G is called *planar* if its vertices and edges can be drawn as points and curves in the plane so that no point lies on the curves, no two curves intersect except for their end points, and no two vertices are drawn at the same point. In such a drawing, the plane is divided into several connected regions, each of which is called a *face*. A face is characterized by the cycle of edges of G that surrounds the region. Such a cycle is called a *facial cycle*.

A *plane embedding* of a planar graph G consists of an ordering of edges around each vertex and the outer face. A planar graph with a fixed plane embedding is called a *plane graph*. Let $f^o(G)$ denote the outer facial cycle of a plane graph G, and let $V^o(G)$ denote $V(f^o(G))$. The set of faces of a plane graph G is denoted by $F(G)$. A vertex (respectively, an edge) in the outer facial cycle is called an *outer vertex* (respectively, an *outer edge*), while a vertex (respectively, an edge) not in the outer facial cycle is called an *inner vertex* (respectively, an *inner edge*).

Let $G = (V, E, F)$ be a biconnected plane graph. A *corner* λ around a vertex v is defined by a pair (v, f) of the vertex v and the facial cycle f whose interior contains the corner. Let $\Lambda(v)$ denote the set of all corners around a vertex v in G, and $\Lambda(G)$ denote the set of all corners in G. A corner (v, f^o) of a vertex v in the outer facial cycle f^o of G is an *outer corner* of f^o. We let $\Lambda^o(G)$ denote the set of the outer corners of the outer facial cycle f^o. We call a cycle C in G a *cut-cycle* if a cut-pair $\{u, v\} \subseteq V(C)$ separates the vertices outside C from those along C (including those inside C). A corner (v, f) of a vertex v in a cut-cycle C is an *outer corner* of C if v is not in the cut-pair of C, and f is one of the two facial cycles outside C that share the cut-pair of C. We denote by $\Lambda^o(C)$ the set of the outer corners of a cut-cycle (or the outer facial cycle) C. For example, $C_1 = (u_6, u_9, u_2, u_{18}, u_8)$ of a graph G in Fig. 1 is a cut-cycle, where $\Lambda^o(C_1) = \{(u_9, f_{12}), (u_{18}, f_9), (u_8, f_9)\}$.

Let $\{u, v\} \in V(C)$ be the cut-pair that separates the vertices outside a cut-cycle C from those along C in G. We consider a subgraph H of G such that the boundary $f^o(H)$ is a cut-cycle in G, where we treat H as a plane graph under the same embedding of G. For such a plane graph H, we define the u, v*-boundary path* $f^o_{uv}(H)$ of H to be the path obtained by traversing the boundary $f^o(H)$ of H from u to v in clockwise order. We denote $V(f^o_{uv}(H)) - \{u, v\}$ by $V^o_{uv}(H)$, and denote the set of outer corners of $f^o_{uv}(H)$ by $\Lambda^o_{uv}(H)$ (i.e., $\Lambda^o_{uv}(H) = \Lambda^o(f^o(H)) \cap (\bigcup_{w\in V^o_{uv}(H)} \Lambda(w))$). We denote $V^o_{uv}(H) \cup V^o_{vu}(H)$ by $V^o(H)$ (note that $u, v \notin V^o(H)$), and $\Lambda^o_{uv}(H) \cup \Lambda^o_{vu}(H)$ by $\Lambda^o(H)$, respectively. For example, cut-cycle $C_1 = (u_6, u_9, u_2, u_{18}, u_8)$ in Fig. 1 has subgraph H_1 with edges $(u_6, u_9), (u_9, u_2), (u_9, u_{10}), (u_{10}, u_6), (u_2, u_{18}), (u_{18}, u_8), (u_8, u_6)$ such that $C_1 = f^o(H_1)$, where $\Lambda^o_{u_6,u_2}(C_1) = \{(u_9, f_{12})\}$ and $\Lambda^o_{u_2,u_6}(C_1) = \{(u_{18}, f_9), (u_8, f_9)\}$ hold.

For a cut-pair $\{u, v\}$ of a biconnected plane graph G, a u, v*-component* H is a connected subgraph of G that either consists of a single edge (u, v) or is a maximal subgraph such that $H - \{u, v\}$ remains connected. We may treat a u, v-component H of a plane graph G as a plane graph under the same embedding of G. In this case, the boundary $f^o(H)$ of H is a cut-cycle. For example, the subgraph H consisting of edges $(u_6, u_9), (u_9, u_2), (u_9, u_{10}), (u_{10}, u_6)$ is a u_6, u_2-component of graph G in Fig. 1. Note that a cut-cycle C is not necessarily the boundary $f^o(H)$ of some u, v-component H. For example, the cut-cycle $C_1 = (u_6, u_9, u_2, u_{18}, u_8)$ in graph G has no such u, v-component H.

A biconnected plane graph G is called *internally triconnected* if, (i) for each inner vertex v with $d_G(v) \geqslant 3$, there exist three paths disjoint except for v, each connecting v and an outer vertex; and (ii) every cycle of G which has no outer edge has at least three vertices v with $d_G(v) \geqslant 3$.

2.2. The SPQR tree of a biconnected planar graph

First we review the definition of *triconnected components* [15]. If G is triconnected, then G itself is the unique triconnected component of G. Otherwise, let $\{u, v\}$ be a cut-pair of G. We split the edges of G into two disjoint subsets E_1 and E_2, such that $|E_1| > 1$, $|E_2| > 1$, and the subgraphs G_1 and G_2 induced by E_1 and E_2 only have vertices u and v in common. Form the graph G'_1 from G_1 by adding an edge (called a *virtual edge*) between u and v that represents the existence of the other subgraph G_2; similarly form G'_2. We continue the splitting process recursively on G'_1 and G'_2. The process stops

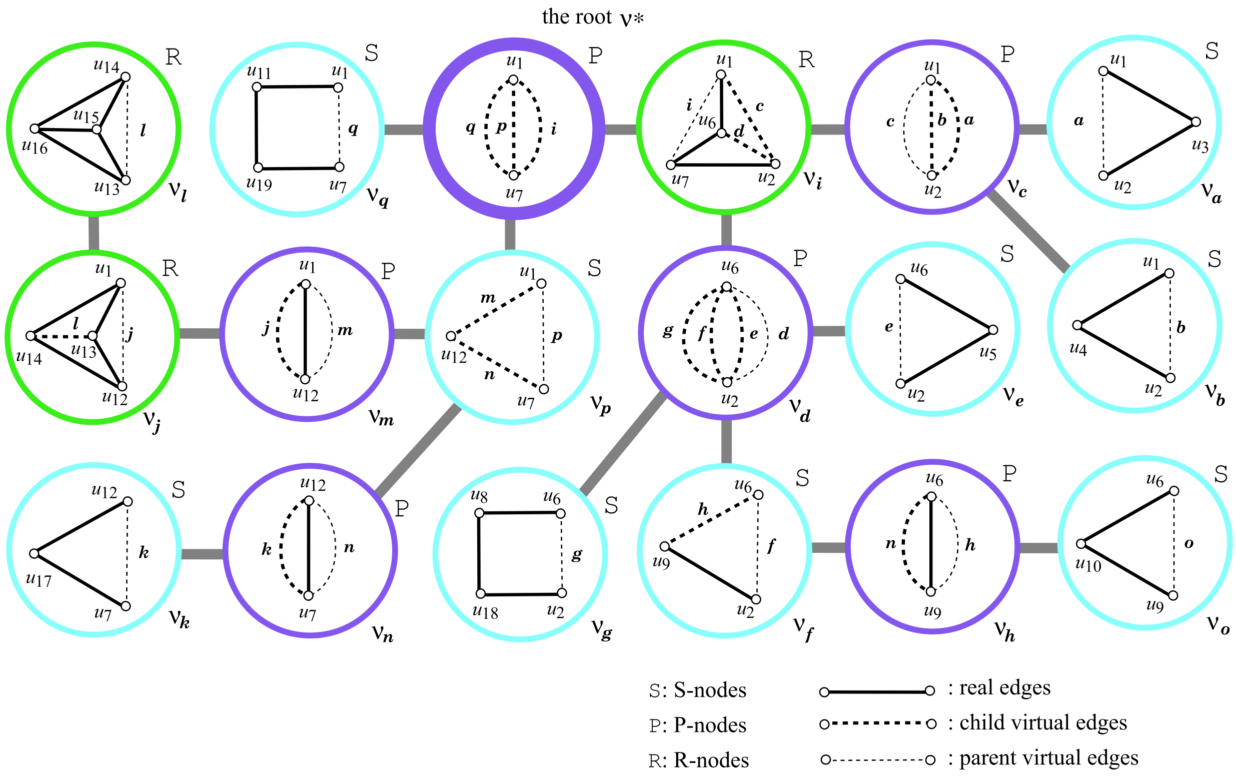

Fig. 2. The SPR tree of the biconnected plane graph G in Fig. 1.

when each resulting graph reaches one of three forms: a triconnected simple graph, a set of three multiple edges (a triple bond), or a cycle of length three (a triangle). The triconnected components of G are obtained from these resulting graphs: (i) a triconnected simple graph; (ii) a *bond*, formed by merging the triple bonds into a maximal set of multiple edges; (iii) a *polygon*, formed by merging the triangles into a maximal simple cycle.

One can define a tree structure, sometimes called the *3-block tree*, using triconnected components as follows. The nodes of the 3-block tree are the triconnected components of G. The edges of the 3-block tree are defined by the virtual edges, that is, if two triconnected components have a virtual edge in common, then the nodes that represent the two triconnected components in the 3-block tree are joined by an edge that represents the virtual edge. There are many variants of the 3-block tree in the literature; the first was defined by Tutte [25]. In this paper, we use the terminology of the *SPQR tree*, a data structure with efficient operations defined by di Battista and Tamassia [6].

Each node ν in the SPQR tree is associated with a graph called the *skeleton* of ν, denoted by $\sigma(\nu) = (V_\nu, E_\nu)$ $(V_\nu \subseteq V)$, which corresponds to a triconnected component. There are four types of nodes in the SPQR tree. The node types and their skeletons are:

1. Q-node: the skeleton consists of two vertices connected by two edges. Each Q-node corresponds to an edge of the original graph.
2. S-node: the skeleton is a simple cycle with at least 3 vertices (this corresponds to a polygon triconnected component).
3. P-node: the skeleton consists of two vertices connected by at least 3 edges (this corresponds to a bond triconnected component).
4. R-node: the skeleton is a triconnected graph with at least 4 vertices.

In fact, we use a slight modification of the SPQR tree: we omit the Q-nodes and we root the tree as described below. We will refer to the (modified) SPQR tree as the SPR tree throughout this paper. The SPR tree is unique, and can be computed in linear-time [6,8,15]. Fig. 2 shows the SPR tree of the biconnected planar graph in Fig. 1.

We treat the SPR tree as a rooted tree $\mathcal{T}$ by choosing a node ν^* as its root. For a node ν, let $\mathrm{Ch}(\nu)$ denote the set of all children of ν, and let η be the parent of ν. The graph $\sigma(\eta)$ has exactly one *virtual edge* e in common with $\sigma(\nu)$. The edge e is called the *parent virtual edge* $\mathrm{parent}(\nu)$ of $\sigma(\nu)$, and a *child virtual edge* of $\sigma(\eta)$. We define the *parent cut-pair* of ν as the two end vertices of $\mathrm{parent}(\nu)$. We denote the graph formed from $\sigma(\nu)$ by deleting its parent virtual edge as $\sigma^-(\nu) = (V_\nu, E^-_\nu)$, $E^-_\nu = E_\nu - \{\mathrm{parent}(\nu)\}$. Let $G^-(\nu)$ denote the subgraph of G which consists of the vertices and real edges in the graphs $\sigma^-(\mu)$ for all descendants μ of ν, including ν itself. For notational convenience, we let $G^-(\nu)$, $\sigma^-(\nu)$

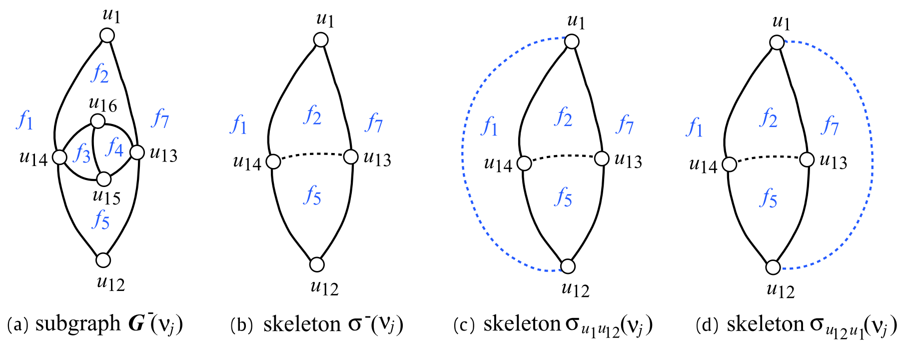

Fig. 3. (a) Subgraph $G^-(\nu_j)$ of R-node ν_j in the SPR tree of G; (b) Skeleton $\sigma^-(\nu_j)$ of R-node ν_j; (c) Skeleton $\sigma_{u_1u_{12}}(\nu_j)$ of R-node ν_j; (d) Skeleton $\sigma_{u_{12}u_1}(\nu_j)$ of R-node ν_j.

and E^-_ν denote $G(\nu)$, $\sigma(\nu)$ and E_ν if ν is the root. By the definition, no S-node (respectively, P-node) has a child S-node (respectively, child P-node), and no P-node can be a leaf in the SPR tree.

Consider the case where G is a plane graph. For a face $f \in F$ of G, we say that a node ν in the SPR tree is *incident* to f if $\sigma(\nu)$ contains the face corresponding to f (note that there may exist more than one such node ν). We choose a node incident to the outer face $f^o(G)$ as the root ν^* of the SPR tree. In particular, we choose a P- or S-node incident to $f^o(G)$ (if any) as the root ν^*, and choose an R-node incident to $f^o(G)$ only when there is no such P- or S-node. Hence, we can assume that if the root ν^* is an R-node, then no outer edge in $\sigma(\nu^*)$ is a virtual edge.

When G is a plane graph, we also treat graphs $\sigma^-(\nu)$ and $G^-(\nu)$ as plane graphs induced from the embedding of G. For a non-root node ν with $(u, v) = \mathrm{parent}(\nu)$, two plane embeddings for $\sigma(\nu)$ can be obtained from the plane graph $\sigma^-(\nu)$ by drawing the parent virtual edge $e = (u, v)$ outside $\sigma^-(\nu)$; one has $f^o_{uv}(\sigma^-(\nu))$ plus e as its boundary, and the other has $f^o_{vu}(\sigma^-(\nu))$ plus e as its boundary, where we denote the former and latter plane graphs by $\sigma_{uv}(\nu)$ and $\sigma_{vu}(\nu)$, respectively. Fig. 3 illustrates examples of $\sigma_{uv}(\nu)$, $\sigma_{vu}(\nu)$ and $G^-(\nu)$ of an R-node ν.

2.3. Straight-line drawings, convex drawings and star-shaped drawings

For two points p_1, p_2 in the plane, $[p_1, p_2]$ denotes the line-segment with end points p_1 and p_2, and for three points p_1, p_2, p_3, $[p_1, p_2, p_3]$ denotes the triangle with three corners p_1, p_2, p_3. The *kernel* $K(P)$ of a polygon P is the set of all points from which all points in P are visible. A polygon is called *star-shaped* if it contains an internal point p^* from which any point p on the boundary of the polygon is visible (i.e., the line-segment $[p^*, p]$ contains no other point on the boundary of the polygon).

A *straight-line drawing* D of a graph $G = (V, E)$ in the plane is an embedding of G in the two dimensional space $\Re^2$, such that each vertex $v \in V$ is drawn as a point $\tau_D(v) \in \Re^2$, and each edge $(u, v) \in E$ is drawn as a straight-line segment $[\tau_D(u), \tau_D(v)]$, where $\Re$ is the set of reals. Let D be a straight-line planar drawing of a biconnected plane graph G. A corner of G is called *concave* in D if its angle in D is greater than π. A vertex v in a straight-line drawing D is called *concave* if one of the corners around v is concave in D. Let $\Lambda^c(D)$ denote the set of all concave corners in D.

A *star-shaped drawing* of a plane graph is a straight-line planar drawing such that each inner facial cycle is drawn as a star-shaped polygon, and the outer facial cycle is drawn as a convex polygon. An outer vertex in a straight-line drawing of a plane graph is called an *apex* if it has a concave corner in the drawing and its concave corner appears in the outer face. Fig. 1(b) shows a star-shaped drawing of the plane graph G in Fig. 1(a), where (u_1, f^o), (u_4, f_{13}), (u_6, f_8), (u_7, f^o), (u_{18}, f_9), (u_{10}, f_{10}), (u_{11}, f^o), (u_{14}, f_1), (u_{15}, f_5), and (u_{17}, f_1) are the concave corners, and u_1, u_7, and u_{11} are the apices.

A straight-line planar drawing D of a plane graph $G = (V, E, F)$ is called a *convex drawing*, if every facial cycle is drawn as a convex polygon. We say that a drawing D of a graph G is *extended* from a drawing D' of a subgraph G' of G, if $\tau_D(v) = \tau_{D'}(v)$ for all $v \in V(G')$. A convex polygon drawn for the outer facial cycle of a biconnected plane graph G can be extended to a convex drawing of G when the following conditions hold.

Theorem 1. *(See Chiba et al. [2] and Thomassen [23].) Let $G = (V, E, F)$ be a biconnected plane graph, let D be a drawing of $f^o(G)$ on a convex polygon P with k sides, and let $Q_1, Q_2, \ldots, Q_k$ be the subpaths of $f^o(G)$, each corresponding to a side of P. Then D can be extended to a convex drawing of G if and only if*:

(i) *G is internally triconnected*; *and*
(ii) *The graph $G - V^o(G)$ has no component H such that all the outer vertices adjacent to vertices in H are contained in a single path Q_i, and there is no inner edge (u, v) whose end vertices are contained in a single path Q_i.*

Chiba et al. [2] gave a linear-time algorithm for constructing a convex drawing of an internally triconnected plane graph which satisfies the conditions in Theorem 1. Based on the algorithm, we show that a specified inner face f containing an

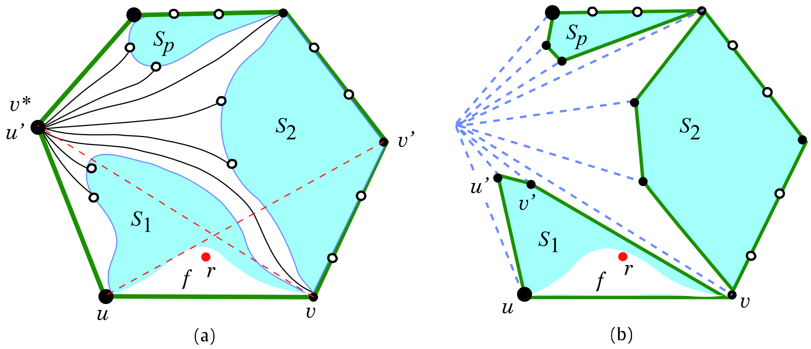

Fig. 4. Illustration of Algorithm CYN(G, S): (a) Apex v^* chosen in (G, S); (b) A sequence of blocks $S_1, S_2, \ldots, S_p$ of $G - v^*$ in Step 2, where apices (respectively, non-apex vertices) are represented by black circles (respectively, white circles).

outer edge (u, v) can be drawn as a sufficiently large convex polygon P_f so that a point r near the edge (u, v) lies in P_f. For a prescribed convex polygon P drawn for the boundary of an internally triconnected plane graph G and an outer edge (u, v), the *safe area* of (u, v) is defined by the intersection of two triangle regions $[u, v, u']$ and $[u, v, v']$, where u' and v' are the closest apices of P to u and v (see Fig. 4(a)).

Lemma 2. *Let $G = (V, E, F)$ be an internally triconnected plane graph, $(u, v) \in E$ be an outer edge and $f \in F$ be the inner face incident to (u, v). Let D be a drawing of $f^o(G)$ on a convex polygon P, where the position of the outer vertices corresponding to non-apices of P are not fixed, and let r be a point in the interior of the safe area of (u, v). Then D can be extended to a convex drawing such that r is contained in the interior of the convex polygon P_f drawn for the face f. Such a drawing D can be found in linear time.*

Proof. A *block* of a graph is a biconnected component, and is called *trivial* if it consists of a single edge. The algorithm reduces the problem of constructing a convex drawing of G into that of constructing convex drawings of several subgraphs of G as follows:

Algorithm CYN(G, P)
Input: An internally triconnected plane graph G, and a convex polygon P drawn for the outer face of G satisfying the conditions in Theorem 1.
Output: A convex drawing of G with the boundary P.
1. Delete an arbitrary apex v^* from G together with the incident edges.
2. Denote by $S_1, S_2, \ldots, S_p$ $(p \geqslant 1)$, the blocks in the resulting graph $G' = G - v^*$ (see Fig. 4(a)).
3. Determine a convex polygon P_i of the outer facial cycle of each block S_i, so that S_i with P_i satisfies the conditions in Theorem 1 (see Fig. 4(b)).
4. Recursively apply the algorithm CYN to each non-trivial block S_i with P_i, in order to determine the positions of vertices not in P_i.

In Step 3, we do not need to determine the positions of non-apex vertices of P_i until they become adjacent to a vertex v^* in the subsequent calls.

To prove the lemma, we choose a vertex v^* and convex polygons P_i more carefully during execution of algorithm CYN(G, P) and its recursive calls. We always choose v^* as u' and choose convex polygons P_i, $i = 1, 2, \ldots, p$, so that the polygon P_j on which edge (u, v) lies contains r in its new safe area of (u, v) (see Fig. 4(b)). Since r is within the safe area of (u, v) in P_j, it is possible to choose such polygon P_j by choosing new apices of P_j as points sufficiently close to v^*. By repeatedly processing the block which contains (u, v) in this way, we can obtain a convex drawing such that the convex polygon P_f drawn for f contains r in its interior. □

3. Configurations and fringe corners

We are now ready to formally define "central edge," "configuration" and "fringe corners." Let G be a biconnected plane graph, and let $\mathcal{T}$ be the rooted SPR tree of G.

For each P-node ν, let $k(\nu)$ denote $|E_\nu^-|$, and for the parent cut-pair $(u, v) = \mathrm{parent}(\nu)$, we number the child edges of ν as $e_1, e_2, \ldots, e_{k(\nu)}$ by traversing these edges from left to right, placing u on the top level and v on the bottom level (see Fig. 5(a)). For each P-node ν, we choose an edge e_{j^*} $(1 \leqslant j^* \leqslant k(\nu))$ of $\sigma^-(\nu)$, which we call the *central edge* of ν and denote it by $c(\nu)$. We always choose the real edge in E_ν^- (if any) as $c(\nu)$. The other virtual edges e_i in $E_\nu^- - \{c(\nu)\}$ are called *left edges* (respectively, *right edges*) if $i < j^*$ (respectively, $i > j^*$). For each child $\mu_i \in \mathrm{Ch}(\nu)$ corresponding to edge e_i, we call subgraph $G^-(\mu_i)$ with $i < j^*$ (respectively, $i > j^*$) a *left component* (respectively, a *right component*) of ν (see Fig. 5(b) and

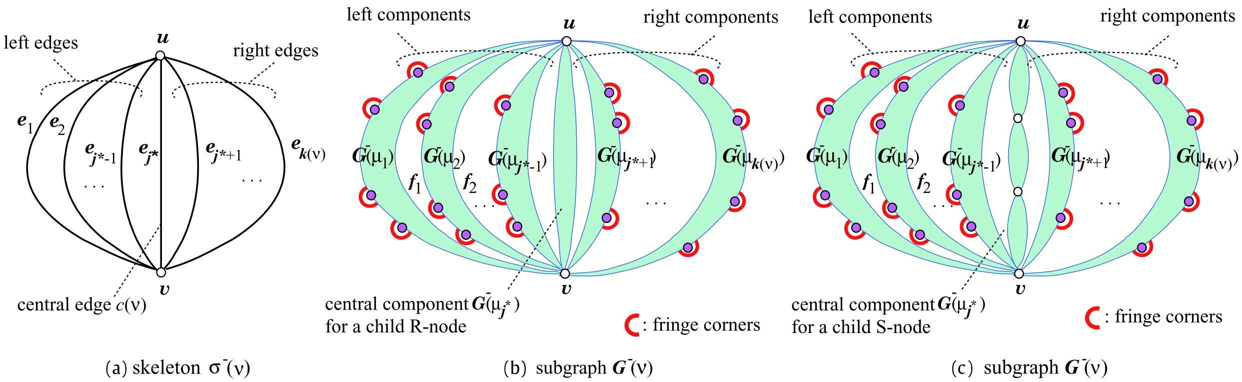

Fig. 5. (a) A plane embedding of skeleton $\sigma^-(\nu)$ of a P-node ν; (b) Fringe corners at P-node ν with a central edge which corresponds to an R-node; (c) Fringe corners at P-node ν with a central edge which corresponds to an S-node.

Fig. 5(c)). If e_{j^*} is the real edge in $\sigma^-(\nu)$, then the edge is called the *central component* of ν; otherwise subgraph $G^-(\mu_{j^*})$ is called the *central component* of ν.

A *configuration* ψ of a rooted SPR tree $\mathcal{T}$ is a set of central edges $c(\nu)$ for all P-nodes ν in $\mathcal{T}$. In this paper, we consider only a straight-line drawing wherein the central component of each P-node ν is drawn so that the drawing can contain the line-segment $[u, v]$ (when we draw the line-segment in the drawing). We call such a drawing *proper*.

Hence, at least one outer corner of the left side of each left component must be chosen as a concave corner to obtain a proper straight-line drawing. Similarly for the concave corners from the right components. Also, in order to obtain any straight-line drawing, we always need to choose three concave corners from the boundary of G, and one concave corner from the boundary $\Lambda^o(G^-(\eta))$ of each R-node η. The set of *fringe corners* for a given configuration ψ formally define necessary choices of concave corners as follows.

Definition 3. Let G be a biconnected plane graph with a SPR tree $\mathcal{T}$, and let ψ be a configuration. We define a set of fringe corners for the root node, each R-node, and each S-node that corresponds to a non-central edge in the skeleton of its parent P-node.

(i) For the root ν^* of $\mathcal{T}$, each outer corner in $\Lambda^o(G)$ is called a *fringe corner* at ν^*. Let $\Lambda^{\mathrm{fr}}(\nu^*) = \Lambda^o(G)$ denote the set of fringe corners at ν^*;
(ii) For each non-root R-node ν that is a child of an R- or S-node, or corresponds to the central edge of a P-node in $\mathcal{T}$, each outer corner in $\Lambda^o(G^-(\nu))$ is called a *fringe corner* at ν. Let $\Lambda^{\mathrm{fr}}(\nu) = \Lambda^o(G^-(\nu))$ denote the set of fringe corners at ν (recall that $\Lambda^o(G^-(\nu)) \cap \Lambda(u) = \Lambda^o(G^-(\nu)) \cap \Lambda(v) = \emptyset$ for $(u, v) = \mathrm{parent}(\nu)$); and
(iii) For each P-node ν, where $(u, v) = \mathrm{parent}(\nu)$ in $\mathcal{T}$, each outer corner in $\Lambda^o_{vu}(G^-(\mu))$ of a left component $G^-(\mu)$ (respectively, $\Lambda^o_{uv}(G^-(\mu))$ of a right component $G^-(\mu)$) is called a *fringe corner* along μ at ν. Let $\Lambda^{\mathrm{fr}}(\mu)$ denote the set of fringe corners along μ at ν (see Fig. 5(b) and Fig. 5(c)).

Note that for a non-root S-node ν which is a child of an R-node, or corresponds to the central edge of a P-node, concave corners on the boundary of a drawing of $G^-(\nu)$ will be provided from the set of fringe corners of its child R- and P-nodes.

Let $\mathcal{L}^{\mathrm{fr}}(\psi)$ denote the family of the sets of fringe corners; i.e., $\mathcal{L}^{\mathrm{fr}}(\psi) = \{\Lambda^{\mathrm{fr}}(\nu^*)\} \cup \{\Lambda^{\mathrm{fr}}(\nu) \mid \nu \text{ is an R-node}\} \cup \{\Lambda^{\mathrm{fr}}(\mu) \mid \mu$ is an S-node corresponding to a non-central edge of its parent P-node$\}$. We call a subset A of fringe corners *proper* if

$$A \cap \Lambda^{\mathrm{fr}} \neq \emptyset \quad \text{for all sets } \Lambda^{\mathrm{fr}} \in \mathcal{L}^{\mathrm{fr}}(\psi), \quad \text{and} \quad \left|A \cap \Lambda^{\mathrm{fr}}(\nu^*)\right| \geqslant 3. \tag{1}$$

We easily see that A is necessary to be proper if a given plane graph G with a configuration ψ has a proper straight-line drawing using the corners in A as concave corners. The main contribution of this paper is to show that the converse is true; i.e., there exists a proper straight-line drawing using the corners in a proper set A as concave corners.

Assume that there is a proper straight-line drawing D of G such that $\Lambda^c(D) = A$ for a proper set A of fringe corners. For the root ν^*, the number $\alpha(\nu^*)$ of apices on the boundary of G is determined by $|A \cap \Lambda^o(G)|$. For a non-root node ν with $(u, v) = \mathrm{parent}(\nu)$, the number $\alpha_{vu}(\nu)$ (respectively, $\alpha_{uv}(\nu)$) of apices on the boundary $f^o_{vu}(G^-(\nu))$ of $G^-(\nu)$ from v to u (respectively, $f^o_{uv}(G^-(\nu))$ of $G^-(\nu)$ from u to v) is determined by $|A \cap \Lambda^o_{vu}(G^-(\nu))|$ (respectively, $|A \cap \Lambda^o_{uv}(G^-(\nu))|$).

Note that we can compute $\alpha_{vu}(\nu)$ and $\alpha_{uv}(\nu)$ for all nodes ν in linear time, as we can compute $\alpha_{vu}(\nu)$ and $\alpha_{uv}(\nu)$ in $O(|V_\nu| + |E_\nu|)$ time from $\alpha_{v'u'}(\mu)$ and $\alpha_{u'v'}(\mu)$, $\mu \in \mathrm{Ch}(\nu)$, where $(u', v') = \mathrm{parent}(\mu)$.

Since A is proper, we see that for the root node ν^*,

$$\alpha(\nu^*) \geqslant 3 \tag{2}$$

by $|A \cap \Lambda^{\mathrm{fr}}(\nu^*)| \geqslant 3$ in (1).

For all non-root R- and S-nodes ν that correspond to left (respectively, right) edges of their parent P-nodes, it holds

$$\alpha_{vu}(\nu) \geqslant 1 \quad (\text{respectively}, \alpha_{uv}(\nu) \geqslant 1) \quad \text{for } (u, v) = \text{parent}(\nu), \tag{3}$$

by (1) and Definition 3(iii).

For all the other non-root S-nodes ν and R-nodes ν (i.e., ν is a child node of an R- or S-node, or corresponds to the central edge of its parent P-node), it holds

$$\alpha_{vu}(\nu) + \alpha_{uv}(\nu) \geqslant 1 \quad \text{for } (u, v) = \text{parent}(\nu), \tag{4}$$

by (1) and Definition 3(ii).

Moreover, each fringe corner in a proper set A is counted at least once in one of the α in (2), (3) and (4). This fact will be used to show that all the fringe corners are used as concave corners in a proper straight-line drawing of G in the next section. Also, a proper set A contains no corners from

$$\Lambda^o\big(\sigma_{uv}(\nu)\big) \quad \big(\text{respectively}, \Lambda^o\big(\sigma_{vu}(\nu)\big)\big) \tag{5}$$

for the skeletons $\sigma(\nu)$ of R- or S-nodes ν with $(u, v) = \text{parent}(\nu)$ which correspond to left (respectively, right) edges of P-nodes, and from

$$\Lambda^o\big(\sigma^-(\nu)\big) \tag{6}$$

for the skeletons $\sigma(\nu)$ of S-nodes ν which correspond to the central edges of P-nodes.

We say that a convex $\alpha(\nu^*)$-gon is *feasible* to the root ν^*. For a non-root node ν with $(u, v) = \text{parent}(\nu)$, we say that a convex $(\alpha_{vu}(\nu) + 2)$-gon (respectively, $(\alpha_{uv}(\nu) + 2)$-gon) is *feasible* to $f^o_{vu}(G^-(\nu))$ (respectively, $f^o_{uv}(G^-(\nu))$), where the two extra apices correspond to outer corners λ_u and λ_v of $G^-(\nu)$ at u and v, respectively. For a non-root node ν with $(u, v) = \text{parent}(\nu)$, we define $\alpha(\nu) = \alpha_{vu}(\nu) + \alpha_{uv}(\nu)$, and say that a convex $(\alpha(\nu) + 2)$-gon B_ν is *feasible* to a non-root node ν if B_ν is obtained by combining a convex $(\alpha_{vu}(\nu)+2)$-gon B_{vu}, feasible to $f^o_{vu}(G^-(\nu))$, and a convex $(\alpha_{uv}(\nu)+2)$-gon B_{uv}, feasible to $f^o_{uv}(G^-(\nu))$. As we will see the details in the next section, for a non-root S-node ν with $(u, v) = \text{parent}(\nu)$, a line-segment $[u, v]$ joining the points drawn for the vertices u and v is called *feasible* to ν.

4. Constructing star-shaped drawings

In this section, we prove that, given a plane graph G with a configuration ψ and a proper subset A of fringe corners, there exists a star-shaped drawing D such that $\Lambda^c(D) = A$, and present a linear-time algorithm to construct such a drawing. The following theorem states the main result of this paper.

Theorem 4. *Let G be a biconnected plane graph, and let ψ be a configuration of G. Then if A is a proper subset of fringe corners, then any convex polygon P with $\Lambda^c(P) = A \cap \Lambda^o(G)$ drawn for $f^o(G)$ can be extended to a star-shaped drawing D of G such that $\Lambda^c(D) = A$. Such a drawing D can be constructed in linear time.*

To prove Theorem 4, we present a divide-and-conquer algorithm that computes a star-shaped drawing for a given proper subset A of fringe corners in a top-down manner along the SPR tree $\mathcal{T}$. Let ν^* be the root of the SPR tree of G. Before we give a full description of the algorithm, we first describe the main ideas to construct a star-shaped drawing recursively:

- We first draw the boundary B_{ν^*} of the root node ν^*, as the prescribed convex polygon P. We then process all the nodes ν in $\mathcal{T}$ from the root to the leaves by repeatedly choosing a feasible convex polygon B_ν, and computing a drawing D_ν of skeleton $\sigma^-(\nu)$ (or $\sigma(\nu)$ for some cases). Note that the virtual edges in D_ν will be replaced with convex drawings of $\sigma^-(\mu)$ or feasible convex polygons of B_μ of corresponding child nodes μ.
- We process each node ν based on its node-type. Roughly speaking, for R-nodes, the main task is to construct a convex drawing of its skeleton and to choose an arbitrary point r_f inside each new face f as the *view point* of f, which will be kept as a visible point in the kernel of the face f until a final drawing D of G is obtained. For P-nodes and S-nodes, the main task is to determine feasible convex polygons and view points accordingly for their children μ, where a view point of an inner face of the skeleton of a child R-node may be specified before constructing a convex drawing of its skeleton in order to apply Lemma 2.

Given a biconnected plane graph G, a configuration ψ, a proper subset of fringe corners A and a prescribed convex polygon P, Algorithm StarShapeDraw first constructs the SPR tree with a root ν^*, and adds an entry $P(\nu^*)$ with $B_{\nu^*} := P$ to $\mathcal{P}$, a set of unprocessed entries. Then it recursively processes each entry $P(\nu)$ from $\mathcal{P}$, by applying Procedures Snode, Rnode or Pnode according to the node type of ν, and adding $P(\mu)$ of each child node $\mu \in \text{Ch}(\nu)$ to $\mathcal{P}$. More formally, Algorithm StarShapeDraw can be described as follows.

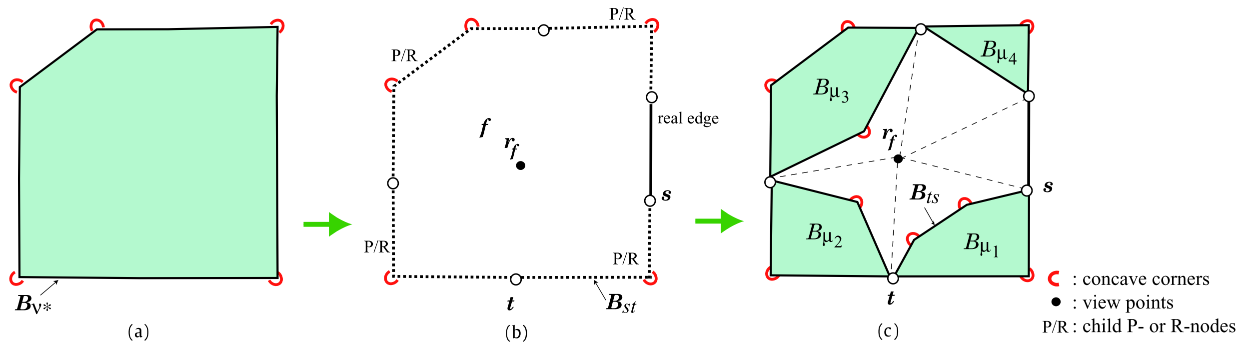

Fig. 6. (a) Input of the root S-node ν^*: feasible convex polygon $B_{\nu^*} = P$; (b) Fixing the boundary B_{ν^*} of the root S-node ν^*; (c) Defining feasible convex polygons B_{μ_i} of the child nodes $\mu_i \in \mathrm{Ch}(\nu^*)$.

Algorithm StarShapeDraw(G, ψ, A, P)
Input: G: biconnected plane graph; ψ: configuration;
A: proper subset of fringe corners; P: convex $|A \cap \Lambda^o(G)|$-gon.
Output: A star-shaped drawing D of G with $\Lambda^c(D) = A$ as an extension of P.

```
Construct the SPR tree T of G;
Choose the root node ν* and make T a rooted tree at ν*;
Compute α_vu(ν) and α_uv(ν) for each node ν;
B_ν* := P;
if ν* is an S-node then
  Choose an arbitrary point r_f within B_ν* as the view point of the inner face f of σ(ν*);
  P(ν*) := (B_ν*, r_f, ∅)
else P(ν*) := (B_ν*, ∅, ∅) /* ν* is an R- or P-node */
end if;
𝒫 := {P(ν*)}; /* Initialisation */
while 𝒫 ≠ ∅ do
   Remove P(ν) from 𝒫;
   if ν is an S-node then SNode(P(ν))
   else if ν is a P-node then PNode(P(ν))
   else RNode(P(ν)) /* compute P(μ) for all μ ∈ Ch(ν) */
   end if;
   𝒫 := 𝒫 ∪ {P(μ) | μ ∈ Ch(ν)}
end while.
```

We now present Procedures SNode, RNode and PNode below, with detailed descriptions of input and output, their types and the main tasks. When a vertex u is drawn as a point in the plane, the point may be denoted by u for notational simplicity. For a subset A of corners, let $V(A)$ denote the set of vertices v such that $\Lambda(v) \cap A \neq \emptyset$.

The main task of Procedure SNode is to compute a feasible convex polygon B_μ to each child node $\mu \in \mathrm{Ch}(\nu)$. More specifically, an S-node ν has the following two types, depending on the input convex polygon.

type I S-node: An S-node ν is called *type I*, if it is the root node or a non-root node, where the corresponding virtual edge e_ν in $\sigma^-(\eta)$ of the parent node η is an *outer* virtual edge in $\sigma^-(\eta)$.

Input: A feasible convex polygon B_ν for $f^o_{vu}(G^-(\nu))$ or $f^o_{uv}(G^-(\nu))$ (or $B_\nu = P$ if $\nu = \nu^*$), and a view point r_1 of an inner face f_1 within B_ν.

Task: We divide the region inside B_ν in a radial manner centered at r_1, and choose a convex polygon B_μ of each child node μ with one of the resulting regions. More specifically, for each child R- or P-node $\mu \in \mathrm{Ch}(\nu)$, we place the corresponding virtual edge (s,t) on B_ν and fix the boundary of $G^-(\mu)$ as a feasible convex $(\alpha(\mu)+2)$-gon B_μ, by combining the series of line-segments of B_ν between s and t, and a new line-segment $[s,t]$ inside B_ν. See the root S-node ν in Fig. 6, and a non-root S-node ν in Fig. 7(a)–(c).

type II S-node: An S-node ν is called *type II*, if it is a non-root node such that the corresponding virtual edge e_ν in $\sigma^-(\eta)$ of the parent node η is an *inner* virtual edge in $\sigma^-(\eta)$.

Input: A feasible polygon B_ν given as a line-segment $[u,v]$ drawn for parent$(\nu) = (u,v)$, and two view points r_1 and r_2 given for two inner faces f_1 and f_2 incident to e_ν.

Task: We divide the union $[u,v,r_1] \cup [u,v,r_2]$ of two triangle regions in a radial manner centered at r_1 and r_2, and choose a feasible convex polygon B_μ of each child node μ with one of the resulting regions. More specifically, for

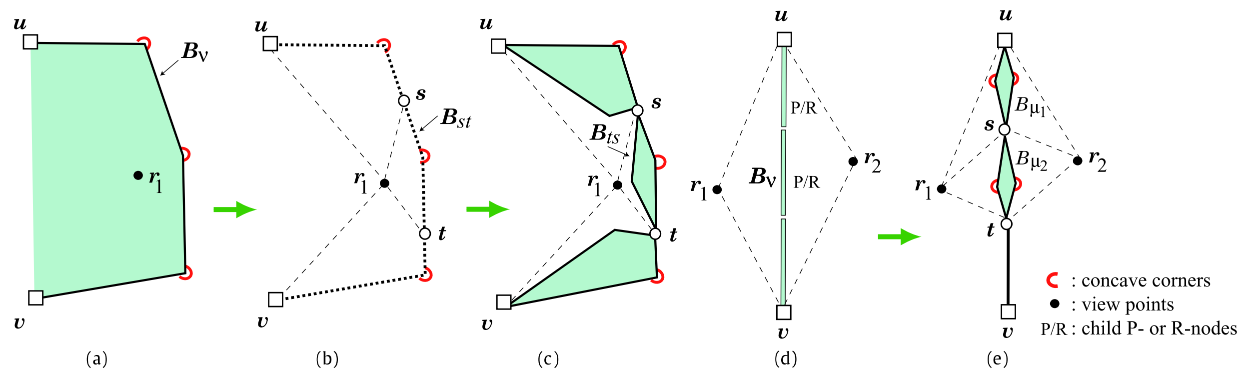

Fig. 7. (a) Input of a non-root type I S-node ν: feasible convex polygon B_ν and a view point r_1; (b) Dividing the region inside B_ν using the view point r_1; (c) Defining feasible convex polygons B_{μ_i} for child nodes $\mu_i \in \mathrm{Ch}(\nu)$ of a non-root type I S-node ν; (d) Input of a non-root type II S-node ν: feasible convex polygon B_ν given as a line-segment $[u, v]$, and two view points r_1 and r_2; (e) Defining feasible convex polygons B_{μ_i} for child nodes $\mu_i \in \mathrm{Ch}(\nu)$ of a type II S-node ν.

each child $\mu \in \mathrm{Ch}(\nu)$, we place the parent cut-pair (s, t) on B_ν, and define the boundary of $G^-(\mu)$ as a feasible convex $(\alpha(\mu)+2)$-gon B_μ inside region $[s, t, r_1] \cup [s, t, r_2]$. See Fig. 7(d) and (e).

The procedure Snode can be formally described as follows.

Procedure Snode($P(\nu)$)
Input: $P(\nu) = (B_\nu, r_1, r_2)$;
(i) type I: a feasible convex polygon B_ν for $f^o_{vu}(G^-(\nu))$ or $f^o_{uv}(G^-(\nu))$ (or $B_\nu = P$ if $\nu = \nu^*$), and a view point r_1.
(ii) type II: a feasible polygon B_ν given as a line-segment $[u, v]$, and two view points r_1 and r_2.
Output: $P(\mu)$ for all child nodes $\mu \in \mathrm{Ch}(\nu)$.

Fix the positions of the remaining vertices in $V_\nu - \{u, v\} - V(A)$ on B_ν;
for each virtual edge $e = (s, t) \in E^-_\nu$ and its corresponding child node $\mu \in \mathrm{Ch}(\nu)$ **do**
  if $r_2 = \emptyset$ **then** /* ν is of type I */
    Let B_{st} be the boundary of B_ν from s to t, where s and t are assumed to appear in this order along the virtual edge (s, t) on B_ν in clockwise order without loss of generality;
    Choose a feasible convex $(\alpha_{ts}(\mu)+2)$-gon B_{ts} for $f^o_{ts}(G^-(\mu))$ inside region $[s, t, r_1]$ such that the union B_μ of B_{st} and B_{ts} is a feasible convex $(\alpha(\mu)+2)$-gon to μ
  else /* ν is of type II */
    Choose a feasible convex $(\alpha_{ts}(\mu)+2)$-gon B_{ts} inside region $[s, t, r_1]$ and a feasible convex $(\alpha_{st}(\mu)+2)$-gon B_{st} inside region $[s, t, r_2]$ such that the union B_μ of B_{ts} and B_{st} is a feasible convex $(\alpha(\mu)+2)$-gon to μ
  end if;
  $P(\mu) := (B_\mu, \emptyset, \emptyset)$
end for.

The main task of Procedure Rnode is first to compute a convex drawing D_ν of $\sigma_{vu}(\nu)$, $\sigma_{uv}(\nu)$ or $\sigma^-(\nu)$ as an extension of B_ν using the linear-time algorithm of Chiba et al. [2]. Then it chooses a feasible convex polygon B_μ for each child node $\mu \in \mathrm{Ch}(\nu)$, and replaces the virtual edges in D_ν with these convex polygons B_μ, $\mu \in \mathrm{Ch}(\nu)$. More specifically, an R-node ν has the following two types, depending on whether we draw $\sigma^-(\nu)$ or $\sigma(\nu)$:

type I R-node: A non-root R-node ν with $(u, v) = \mathrm{parent}(\nu)$ is called *type I*, if it corresponds to a non-central virtual edge e_ν in the skeleton $\sigma^-(\eta)$ of its parent P-node η. Let $\sigma^A_{vu}(\nu)$ (respectively, $\sigma^A_{uv}(\nu)$) denote the graph obtained from $\sigma_{vu}(\nu)$ (respectively, $\sigma_{uv}(\nu)$) by adding the vertices of $V(A \cap \Lambda^o_{vu}(G^-(\nu)))$ (respectively, $V(A \cap \Lambda^o_{uv}(G^-(\nu))))$ on the corresponding virtual edges.

Input: A feasible convex $(\alpha_{vu}(\nu)+2)$-gon (respectively, $(\alpha_{uv}(\nu)+2)$-gon) B_ν for $f^o_{vu}(G^-(\nu))$ (respectively, $f^o_{uv}(G^-(\nu))$), and a view point r_{f^*} of the inner face f^* of $\sigma^A_{vu}(\nu)$ (respectively, $\sigma^A_{uv}(\nu)$) incident to (u, v) (see Fig. 8(a)).

Task: We first construct a convex drawing of $\sigma^A_{vu}(\nu)$ (respectively, $\sigma^A_{uv}(\nu)$), $(u, v) = \mathrm{parent}(\nu)$, within region B_ν so that the convex polygon of face f^* contains r_{f^*} in its interior, and then replace the virtual edge corresponding to each child node μ with a feasible convex polygon B_μ. More specifically, we first extend B_ν to a convex drawing of $\sigma^A_{vu}(\nu)$ (respectively, $\sigma^A_{uv}(\nu)$) such that r_{f^*} lies within the polygon for the face f^* using Lemma 2 (see Fig. 8(b)). Then we choose view points r_f of all inner faces f (other than f^*) in the drawing, and define feasible convex polygons B_μ of $G^-(\mu)$ for all child nodes $\mu \in \mathrm{Ch}(\nu)$ (see Fig. 8(c)).

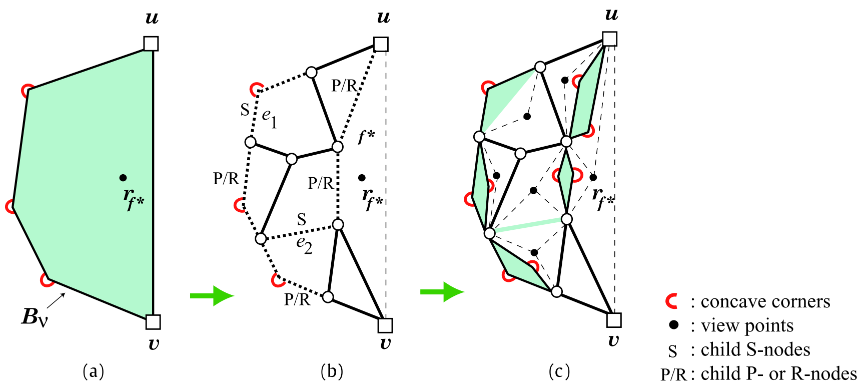

Fig. 8. (a) Input of a type I R-node ν: a feasible convex $(\alpha_{vu}(\nu)+2)$-gon B_ν for $f^o_{vu}(\nu)$ and a view point r_{f^*}; (b) Extending B_ν into a convex drawing of $\sigma_{vu}(\nu)$; (c) Defining feasible convex polygons for child P-, R- and S-nodes $\mu \in \mathrm{Ch}(\nu)$.

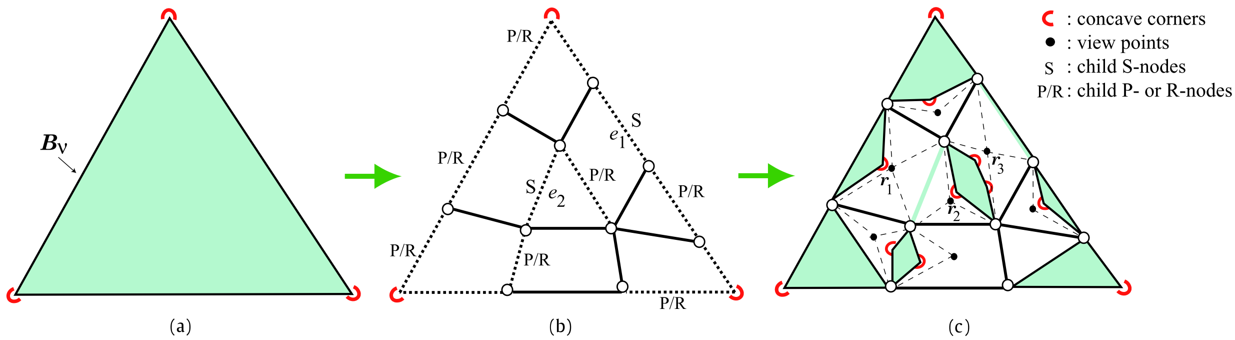

Fig. 9. (a) Input of the root R-node ν^*: feasible convex polygon $B_{\nu^*} = P$; (b) Extending B_{ν^*} into a convex drawing of $\sigma^A(\nu^*)$; (c) Defining feasible convex polygons B_μ of the child P-, R- and S-nodes $\mu \in \mathrm{Ch}(\nu^*)$.

type II R-node: An R-node ν is called *type II*, if it is the root node ν^*, a child node of an S- or R-node, or it corresponds to the central virtual edge of the skeleton $\sigma^-(\eta)$ of its parent P-node η.

Input: A feasible convex $(\alpha(\nu)+2)$-gon (respectively, $\alpha(\nu)$-gon) B_ν to ν (respectively, ν^*) (see Fig. 9(a) and Fig. 10(a)).

Task: We first construct a convex drawing of $\sigma^-(\nu)$ or $\sigma^-(\nu^*)$ within region B_ν, and then replace the virtual edge corresponding to each child node μ with a feasible convex polygon B_μ of μ. More specifically, we compute a convex drawing of $\sigma^A(\nu)$, where $\sigma^A(\nu)$ denotes the graph obtained from $\sigma^-(\nu)$ (respectively, $\sigma(\nu^*)$) by adding the vertices in $V(A \cap \Lambda^o(G^-(\nu)))$ (respectively, $V(A \cap \Lambda^o(G))$) on the corresponding virtual edges. We extend B_ν to a convex drawing of $\sigma^A(\nu)$ using the algorithm of Chiba et al. [2] (see Fig. 9(b) and Fig. 10(b)). Then we choose view points r_f of all inner faces f in the drawing, and fix feasible convex polygons B_μ of $G^-(\mu)$ for all child nodes $\mu \in \mathrm{Ch}(\nu)$ (see Fig. 9(c) and Fig. 10(c)).

The procedure RNODE can be formally described as follows.

Procedure RNODE($P(\nu)$)
Input: $P(\nu) = (B_\nu, r_{f^*}, \emptyset)$;
(i) type I: a feasible convex $(\alpha_{vu}(\nu)+2)$-gon B_ν for $f^o_{vu}(G(\nu))$ (or $(\alpha_{uv}(\nu)+2)$-gon for $f^o_{uv}(G(\nu))$), and a view point r_{f^*}.
(ii) type II: a feasible convex $(\alpha(\nu)+2)$-gon B_ν for $G^-(\nu)$ (or $\alpha(\nu)$-gon for G, if $\nu = \nu^*$).
Output: Convex drawing of H (defined below) and $P(\mu)$ for all child nodes $\mu \in \mathrm{Ch}(\nu)$.

if $r_{f^*} \neq \emptyset$ **then** /* ν is of type I */
  $H := \sigma^A_{vu}(\nu)$ or $\sigma^A_{uv}(\nu)$ for $(u, v) = \mathrm{parent}(\nu)$;
  Extend B_ν to a convex drawing D_ν of H using Lemma 2 so that r_{f^*} is contained in the convex polygon drawn for the face f^*
else /* ν is of type II */
  $H := \sigma^A(\nu)$;
  Extend B_ν to a convex drawing D_ν of H using the algorithm of Chiba et al [2].
end if;
Choose a point r_f in each inner face f of D_ν as the view point of f, where we use r_{f^*} for the face f^* if ν is of type I;
Fix the positions of remaining vertices in $V_\nu - V(A)$ in B_ν;

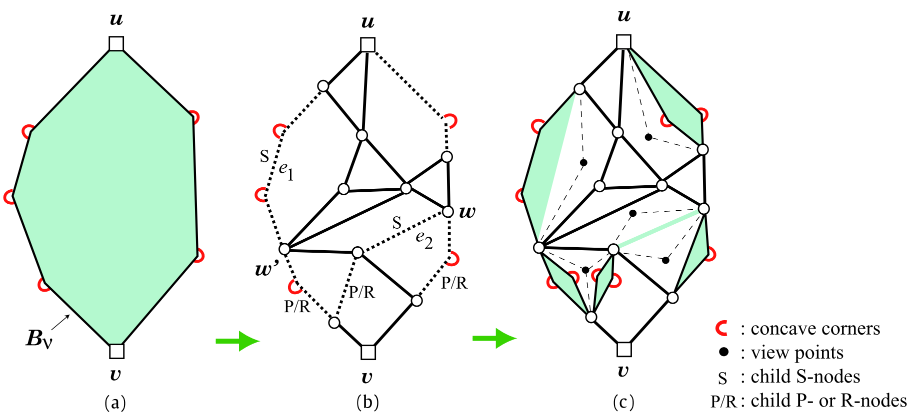

Fig. 10. (a) Input of a type II R-node ν: a feasible convex polygon B_ν of ν; (b) Extending B_ν into a convex drawing of $\sigma^A(\nu)$; (c) Defining feasible convex polygons B_μ of the child P-, R- and S-nodes $\mu \in \mathrm{Ch}(\nu)$.

for each *outer* virtual edge $e = (s,t)$ in H **do**
  if e corresponds to a child R- or P-node $\mu \in \mathrm{Ch}(\nu)$ **then**
    Fix the boundary B_μ of $G^-(\mu)$ as a feasible convex $(\alpha(\mu)+2)$-gon, as in line 5 of SNODE for type I S-nodes;
    $P(\mu) := (B_\mu, \emptyset, \emptyset)$
  else /* e corresponds to a child S-node $\mu \in \mathrm{Ch}(\nu)$ */
    Let B_{st} be the boundary of B_ν from s to t, where s and t are assumed to appear in this order along the virtual edge (s,t) on B_ν in clockwise order without loss of generality;
    Let r_f be the view point of the inner face f of B_ν incident to e;
    $P(\mu) := (B_\mu := B_{st}, r_f, \emptyset)$
  end if
end for;
for each *inner* virtual edge $e = (s,t)$ in H **do**
  if e corresponds to a child R- or P-node $\mu \in \mathrm{Ch}(\nu)$ **then**
    Fix the boundary B_μ of $G^-(\mu)$ as a feasible convex $(\alpha(\mu)+2)$-gon, as in line 7 of SNODE for type II S-nodes;
    $P(\mu) := (B_\mu, \emptyset, \emptyset)$
  else /* e corresponds to a child S-node $\mu \in \mathrm{Ch}(\nu)$ */
    Let B_μ be the line-segment $[s,t]$;
    Let r_{f_1} and r_{f_2} be the view points of the two inner faces f_1 and f_2 of B_ν incident to e;
    $P(\mu) := (B_\mu, r_{f_1}, r_{f_2})$
  end if
end for.

The main task of Procedure PNODE is to fix the boundary of $G^-(\mu)$ for each child node $\mu \in \mathrm{Ch}(\nu)$ as a feasible convex $(\alpha_{vu}(\mu)+2)$-, $(\alpha_{uv}(\mu)+2)$- or $(\alpha(\mu)+2)$-gon B_μ if μ corresponds to a left, right or the central edge of ν with respect to configuration ψ, respectively.

P-node: Let $V_\nu = \{u, v\}$ for a P-node ν, and let $(e_1, e_2, \ldots, e_{j^*}, \ldots, e_k)$ denote the sequence of the edges in $\sigma^-(\nu)$, where $e_{j^*} = c(\nu)$ is the central edge, and f_i denotes the face between two edges e_i and e_{i+1} in the plane graph $\sigma^-(\nu)$ (see Fig. 11(a) and Fig. 12(a)).

Input: A feasible convex $(\alpha(\nu)+2)$-gon B_ν for a non-root P-node ν (respectively, $\alpha(\nu^*)$-gon for the root P-node ν^*).

Task: We replace each virtual edge with a feasible convex polygon B_μ, corresponding to a left, right, or the central edge of ν with respect to configuration ψ. More specifically, we choose a feasible convex polygon B_μ for each child node $\mu \in \mathrm{Ch}(\nu)$ and view points accordingly, by processing left edges in E_ν^- from e_1 to e_{j^*-1}, right edges in E_ν^- from e_k to e_{j^*+1}, and the central edge e_{j^*} as follows (see Fig. 11(b) and Fig. 12(b)):

Left edges: If the left edge e_i corresponds to an S-node (respectively, R-node) μ_i, then we choose a view point r_{f_i} and treat μ_i as a type I S-node (respectively, type I R-node).

Right edges: We apply the above procedure symmetrically to right edges $e_k, e_{k-1}, \ldots, e_{j^*+1}$.

Central edges:

- If the central edge e_{j^*} with $1 < j^* < k$ corresponds to an S-node (respectively, R-node) μ_{j^*}, then we treat μ_{j^*} as a type II S-node (respectively, type II R-node).

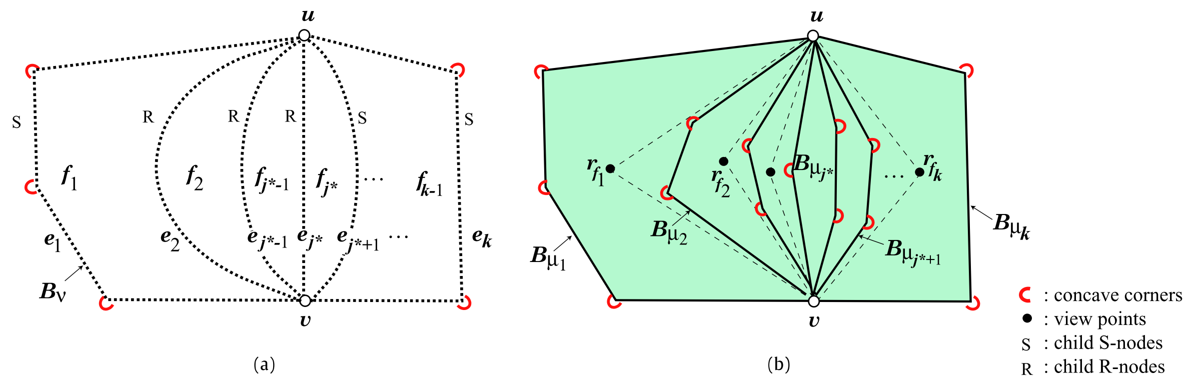

Fig. 11. (a) Input feasible polygon B_ν of a P-node ν with $1 < j^* < k$; (b) feasible convex polygons B_μ for all child nodes $\mu \in \mathrm{Ch}(\nu)$, and view points r_{f_i} of inner faces f_i $(1 \leqslant i \leqslant j^* - 1)$, and $r_{f_{i-1}}$ of inner faces f_i $(j^* + 1 \leqslant i \leqslant k)$.

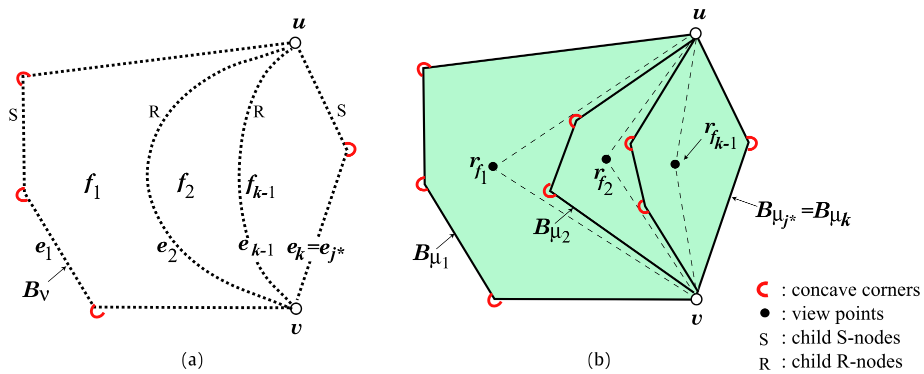

Fig. 12. (a) Input feasible polygon B_ν of a P-node ν with $1 < j^* = k$; (b) feasible convex polygons B_μ for all child nodes $\mu \in \mathrm{Ch}(\nu)$, and view points r_{f_i} of inner faces f_i $(1 \leqslant i \leqslant k - 1 = j^* - 1)$.

- If the central edge e_{j^*} with $j^* \in \{1, k\}$ corresponds to an S-node (respectively, R-node) μ_{j^*}, then we treat μ_{j^*} as a type I S-node (respectively, type I R-node).

The procedure PNODE can be formally described as follows.

Procedure PNODE($P(\nu)$)
Input: $P(\nu) = (B_\nu, \emptyset, \emptyset)$, where B_ν is a feasible convex polygon for $G^-(\nu)$.
Let $V_\nu = \{u, v\}$, and let $(e_1, e_2, \ldots, e_{j^*}, \ldots, e_k)$ denote the sequence of the edges in $\sigma^-(\nu)$, where $e_{j^*} = c(\nu)$ and $\mu_j \in \mathrm{Ch}(\nu)$ denotes the child node corresponding to e_j.
Output: $P(\mu_j)$ for all child nodes $\mu_j \in \mathrm{Ch}(\nu)$.
1: Let B_{vu} (respectively, B_{uv}) be the polygon that consists of the boundary of B_ν from v to u (respectively, from u to v) and line-segment $[u, v]$;
2: **for** $i = 1, 2, \ldots, j^* - 1$ **do** /* lines 2–10: process left child edges */
3: **if** $i = 1$ **then**
4: $B_{\mu_i} := B_{vu}$
5: **else** /* $2 \leqslant i \leqslant j^* - 1$ */
6: Choose a feasible convex $(\alpha_{vu}(\mu_i) + 2)$-gon B_{μ_i} to $f^o_{vu}(G^-(\mu_i))$ within region $[u, v, r_{f_{i-1}}]$
7: **end if**;
8: Choose a view point r_i of f_i in B_{μ_i}, where we choose r_i within the safe area of (u, v) in B_{μ_i} if μ_i is an R-node;
9: $P(\mu_i) := (B_{\mu_i}, r_i, \emptyset)$
10: **end for**;
/* lines 11–16: process the outer central edge: case of $j^* = k$ */
11: **if** $j^* = k$ and e_{j^*} corresponds to a child S-node $\mu_k \in \mathrm{Ch}(\nu)$ **then** $P(\mu_k) := (B_{\mu_k} := B_{uv}, r_{k-1}, \emptyset)$
12: **end if**;
13: **if** $j^* = k$ and e_{j^*} corresponds to a child R-node $\mu_k \in \mathrm{Ch}(\nu)$ **then**
14: Choose a feasible convex $(\alpha(\mu_k) + 2)$-gon B_{μ_k} by combining B_{uv} and a feasible convex $(\alpha_{vu}(\mu_k) + 2)$-gon B'_{vu} to $f^o_{vu}(G^-(\mu_k))$ within $[u, v, r_{k-1}]$;

15: $\quad P(\mu_k) := (B_{\mu_k}, \emptyset, \emptyset)$
16: **end if**;
17: **for** $i = k, k-1, \ldots, j^*+1$ **do** /* process right child edges */
18: $\quad$ Apply the same procedure in lines 2–10 to e_i by exchanging α_{vu} with α_{uv} and B_{vu} with B_{uv}
19: **end for**;
20: Apply the same procedure in lines 11–16 to e_{j^*} with $j^* = 1$ by exchanging α_{vu} with α_{uv} and B_{vu} with B_{uv};
/* lines 21–26: process the inner central edge */
21: **if** e_{j^*} with $1 < j^* < k$ corresponds to a child S-node $\mu_{j^*} \in \mathrm{Ch}(\nu)$ **then**
22: $\quad P(\mu_{j^*}) := (B_{\mu_{j^*}} := [u, v], r_{j^*-1}, r_{j^*})$
23: **end if**;
24: **if** e_{j^*} with $1 < j^* < k$ corresponds to a child R-node $\mu_{j^*} \in \mathrm{Ch}(\nu)$ **then**
25: $\quad$ Choose a feasible convex $(\alpha(\mu_{j^*}) + 2)$-gon $B_{\mu_{j^*}}$ by combining a feasible convex $(\alpha_{vu}(\mu_{j^*}) + 2)$-gon B'_{vu} to $f^o_{vu}(G^-(\mu_{j^*}))$ and a feasible convex $(\alpha_{uv}(\mu_{j^*}) + 2)$-gon B'_{uv} to $f^o_{uv}(G^-(\mu_{j^*}))$ within $[u, v, r_{j^*-1}] \cup [u, v, r_{j^*}]$;
26: $\quad P(\mu_{j^*}) := (B_{\mu_{j^*}}, \emptyset, \emptyset)$
27: **end if**.

Proof of Theorem 4. We prove the correctness of the algorithm StarShapeDraw by showing that feasible polygons B_μ, $\mu \in \mathrm{Ch}(\nu)$ can be obtained from a feasible polygon B_ν by any application of Snode, Rnode and Pnode and that all view points r_f remain visible from any point on the boundaries of the corresponding faces f.

For a node ν and a child node $\mu \in \mathrm{Ch}(\nu)$ in the SPR tree $\mathcal{T}$ of G, we show that the polygon B_μ computed from a feasible polygon B_ν in $P(\nu) = (B_\nu, r_1, r_2)$ by Snode, Rnode and Pnode is feasible.

Case-1. ν is an S-node: Then μ is an R- or P-node, and $\alpha(\mu) \geqslant 1$ holds by (4). Let $(s,t) \in E^-_\nu$ be the virtual edge corresponding to μ.

(i) ν is a type I S-node: Assume that ν is the root node ν^*. Then B_{ν^*} has at least 3 apices since $|A \cap \Lambda^o(G)| = \alpha(\nu^*) \geqslant 3$ holds by (2). Since the view point r_1 of the inner face of $\sigma(\nu^*)$ is chosen within B_{ν^*}, we can choose a convex $(\alpha_{ts}(\mu) + 2)$-gon B_{ts} inside region $[s, t, r_1]$, which is feasible to $f^o_{ts}(G^-(\mu))$, so that B_{ts} together with the boundary B_{st} of B_ν from s to t gives a convex $(\alpha(\mu) + 2)$-gon B_μ, which is feasible to μ. We can choose such B_μ by (3) if we choose B_{ts} sufficiently close to $[s, t]$ so that the inner angles at points s and t are both less than π (see Fig. 6(a)–(c)).

The case where ν is a non-root node can be treated analogously since r_1 is chosen as a point from which the boundary B_{ts} is visible for the case where μ corresponds to the leftmost or rightmost child virtual edge of a P-node (see Fig. 7(c), and $r_{f_{k-1}}$ and B_{μ_k} in Fig. 12(b)).

(ii) ν is a type II S-node: In this case, B_ν is given as a line-segment $[u, v]$. We see by induction of applications of procedures Snode, Rnode and Pnode that the two view points r_1 and r_2 lie on different sides along the straight-line containing $[u, v]$. Hence each region $[s, t, r_i]$, $i = 1, 2$ is a convex 3-gon, and by (4), we can choose a convex $(\alpha(\mu) + 2)$-gon, which is feasible to μ, by combining a convex $(\alpha_{ts}(\mu) + 2)$-gon B_{ts} inside region $[s, t, r_1]$ and a convex $(\alpha_{st}(\mu) + 2)$-gon B_{st} inside region $[s, t, r_2]$ (see Fig. 7(d) and (e)).

Case-2. ν is an R-node: We first show that the graph H can be extended to a convex drawing in lines 1–7 of Rnode.

(i) If ν is a non-root type II R-node with $(u, v) = \mathrm{parent}(\nu)$, then $\sigma^-(\nu)$ is internally triconnected and any cut-pair in $\sigma^-(\nu)$ separates u and v, since $\sigma(\nu)$ is triconnected. If ν is the root R-node or a non-root type II R-node, then the pair of $H = \sigma^A(\nu)$ and B_ν satisfies the condition of Theorem 1, since any vertex added to $\sigma^-(\nu)$ to obtain the plane graph $H = \sigma^A(\nu)$ forms an apex of the boundary B_ν.

(ii) Similarly, if ν is a non-root type I R-node with $(u, v) = \mathrm{parent}(\nu)$, then $\sigma_{vu}(\nu)$ (respectively, $\sigma_{uv}(\nu)$) is triconnected and $H = \sigma^A_{vu}(\nu)$ (respectively, $\sigma^A_{uv}(\nu)$) and B_ν satisfy the condition of Theorem 1. Hence B_ν can be extended to a convex drawing D_ν of H. In particular, for $H = \sigma^A_{vu}(\nu)$ (respectively, $\sigma^A_{uv}(\nu)$), we can find a convex drawing D_ν such that r_{f^*} is contained in the convex polygon drawn for the face f^* by Lemma 2.

We next consider child nodes $\mu \in \mathrm{Ch}(\nu)$ corresponding to outer virtual edges in H. Assume without loss of generality that the vertices s and t in parent(μ) appear in this order along the virtual edge (s, t) on the boundary of H in clockwise order. If ν is the root node ν^*, then $\alpha(\nu^*) = \sum\{\alpha_{st}(\mu) \mid \mu \in \mathrm{Ch}(\nu)$ corresponding to outer edges of $\sigma(\nu^*)\} + |\{v \in V_\nu \mid \lambda(v) \in A \cap \Lambda^o(G)\}|$. If ν is a non-root node with $(u, v) = \mathrm{parent}(\nu)$, then $\alpha(\nu) = \sum\{\alpha_{st}(\mu) \mid \mu \in \mathrm{Ch}(\nu)$ corresponding to outer edges of $\sigma_{vu}(\nu)\} + |\{v \in V_\nu \mid \lambda(v) \in A \cap \Lambda^o_{vu}(G)\}|$. Hence, the boundary B_{st} of B_ν from s to t in lines 12 and 15 of Rnode gives a feasible polygon to $f^o_{st}(G^-(\mu))$. If μ is an S-node, then this gives a feasible polygon to μ. If μ is an R- or P-node, then we see that a feasible polygon B_{ts} can be chosen in line 12 such that the union of B_{st} and B_{ts} gives a feasible polygon to μ by (3) and (4).

We finally consider child nodes $\mu \in \mathrm{Ch}(\nu)$ corresponding to inner virtual edges in $\sigma^-(\nu)$. In this case, each inner virtual edge is drawn as a line-segment $[s, t]$ in the drawing D_ν of H, and a view point r_f is chosen with the convex polygon drawn for each inner face f. If μ is an S-node, then it is easy to see that $B_\mu := [s, t]$ is feasible to μ. If μ is an R- or P-node, then a feasible polygon B_μ to μ can be obtained by choosing feasible polygons B_{ts} and B_{st} within regions $[s, t, r_{f_1}]$ and $[s, t, r_{f_2}]$ for the inner faces f_1 and f_2 incident to edge (s, t), respectively.

Case-3. ν is a P-node with $V_\nu = \{u, v\}$: Let $e_1, \ldots, e_{j^*}, \ldots, e_k$ and $\mu_i \in \mathrm{Ch}(\nu)$ be defined as in the input of Pnode.

We first consider the case where μ is a child R- or S-node μ_j with $j \neq j^*$. If $j = 1$, then μ corresponds to the outer virtual edge e_1 and a feasible polygon B_μ to $f^o_{vu}(G^-(\mu))$ is given by the boundary B_{vu} of B_ν from v to u in line 1 of PNODE, since B_{vu} is feasible to $f^o_{vu}(G^-(\nu)) = f^o_{vu}(G^-(\mu))$ by the feasibility of B_ν.

For $1 < j < j^*$, we can choose a convex $(\alpha_{vu}(\mu_j)+2)$-gon B_{μ_j} of the jth child node $\mu_j \in \mathrm{Ch}(\nu)$, which is feasible to $f^o_{vu}(G^-(\mu_j))$. Note that $\alpha_{vu}(\mu_j) \geqslant 1$ by (3). Since B_{μ_j} $(1 < j < j^*)$ is chosen within region $[u, v, r_{f_{j-1}}]$ for a view point $r_{f_{j-1}}$ chosen within $B_{\mu_{j-1}}$, no two polygons B_{μ_j} and $B_{\mu_{j'}}$ with $1 \leqslant j < j' \leqslant j^*$ overlap each other except at the line-segment $[u, v]$. For $j^* < j \leqslant k$, we can find feasible polygons B_{μ_j} symmetrically.

Consider $\mu = \mu_{j^*} \in \mathrm{Ch}(\nu)$. If $j^* = k$, then a feasible polygon B_{μ_k} to $f^o_{uv}(G^-(\mu_k))$ is given by the boundary B_{uv} of B_ν from u to v, which is feasible to $f^o_{uv}(G^-(\nu)) = f^o_{uv}(G^-(\mu_k))$ by the feasibility of B_ν. For the S-node $\mu_{j^*} = \mu_k$, B_{uv} is feasible, and for the R-node $\mu_{j^*} = \mu_k$, we can find a convex $(\alpha_{vu}(\mu_k)+2)$-gon B'_{vu} within region $[u, v, r_{k-1}]$ such that the union of B_{uv} and B'_{vu} is a convex $(\alpha(\mu_k)+2)$-gon, which is feasible to μ_k. The case of $j^* = 1$ can be treated symmetrically.

Finally consider the case of $\mu = \mu_{j^*}$ with $1 < j^* < k$. In PNODE, view points r_{j^*-1} and r_{j^*} have been chosen from the both sides of region B_ν divided by the line-segment $[u, v]$ when we choose a polygon $B_{\mu_{j^*}}$ in lines 21–27. In this case, it is easy to see that a feasible polygon $B_{\mu_{j^*}}$ to the S- or R-node μ_{j^*} can be chosen within the region $[u, v, r_{j^*-1}] \cup [u, v, r_{j^*}]$.

We have shown that feasible polygons for the child nodes can be obtained from a feasible polygon of its parent. During execution of STARSHAPEDRAW, no corner in sets (5) and (6) has a chance to be chosen as a concave corner in a final drawing. Moreover, such a corner is not contained in any proper set A by definition.

We finally see that any view point r_f remains visible from any other point within the polygon drawn for the corresponding inner face. For the root S-node ν^* and a view point r_f of the inner face f of $\sigma(\nu^*)$, the boundary B_μ of each child node $\mu \in \mathrm{Ch}(\nu)$ is chosen when SNODE is applied to $P(\nu^*)$ and these boundaries remain unchanged until STARSHAPEDRAW outputs a final drawing, wherein r_f is visible from any point on these boundaries. The other cases can be treated analogously to see that view points remain visible from the boundaries drawn for the corresponding inner faces.

We now show that STARSHAPEDRAW can be implemented to run in linear time. As we have observed, $\alpha_{vu}(\nu)$ and $\alpha_{uv}(\nu)$ for all nodes ν $((u, v) = \mathrm{parent}(\nu))$ in the rooted SPR-tree can be computed in linear time. Let $\Delta\alpha_\nu$ denote the number of apices (concave corners) newly introduced on the polygons B_μ of the child nodes $\mu \in \mathrm{Ch}(\nu)$, when the node ν is processed by one of the procedures SNODE$(P(\nu))$, PNODE$(P(\nu))$ and RNODE$(P(\nu))$. To show the linear-time complexity, it suffices to show that each of these procedures can be executed in $O(|V_\nu| + |E_\nu| + \Delta\alpha_\nu)$ time since $\sum_{\nu \in \mathcal{T}}(|V_\nu| + |E_\nu| + \Delta\alpha_\nu) = O(|V| + |E| + |A|)$ holds. For a P-node ν, we easily see that PNODE$(P(\nu))$ can be performed in $O(|V_\nu| + |E_\nu| + \Delta\alpha_\nu)$ time.

For an S- or R-node ν, the crucial part is how to identify which side of B_ν contains a given outer vertex of the skeleton of ν. For this, we maintain polygons with one-dimensional array as follows. When we newly introduce a convex polygon $B_{uv}(\nu)$ (or $B_{vu}(\nu)$) for a node ν, we store the sequence of the sides (line-segments) of the polygon in a one-dimensional array L so that, for the ith apex λ on the boundary, we can access the sides $L[i-1]$ and $L[i]$ that contain λ as their common endpoint in $O(1)$ time. Part of the boundary L may be reused as a feasible polygon $B_{u'v'}(\mu)$ for some descendant of ν during STARSHAPEDRAW. In this case, $B_{u'v'}(\mu)$ is specified by the indices i_μ and j_μ such that the sequence $L[i_\mu], L[i_\mu+1], \ldots, L[j_\mu]$ gives the sides of $B_{u'v'}(\mu)$.

Consider the case where ν is an S-node (the case where ν is an R-node can be treated in a similar way). In line 1 of SNODE, the main task is to identify the side of B_ν that contains each vertex $v \in V_\nu - \{u, v\} - V(A)$, which may not correspond to any concave corner of B_ν (if v has a concave corner on B_ν, then two sides of B_ν contains v as their common endpoint). Without traversing all concave corners on the boundary B_ν, we can find the corresponding side(s) of each vertex $v \in V_\nu - \{u, v\} - V(A)$ in $O(1)$ time as follows. The sequence of the sides of B_ν is now stored in array $L[i_\nu], L[i_\nu+1], \ldots, L[j_\nu]$. For the first outer vertex $v_1 \in V_\nu - \{u, v\}$, let $\mu_1 \in \mathrm{Ch}(\nu)$ be the child node corresponding to edge (u, v_1) if (u, v_1) is a virtual edge (we set $\alpha_{uv_1}(\mu_1) := 0$ if (u, v_1) is a real edge). If $v_1 \notin V(A)$, then the side of B_ν that contains v_1 can be identified as $L[i_\nu + \alpha_{uv_1}(\mu_1)]$ in $O(1)$ time. If $v_1 \in V(A)$, then the two sides of B_ν that contain v_1 as their common endpoint can be identified as $L[i_\nu + \alpha_{uv_1}(\mu_1)]$ and $L[i_\nu + \alpha_{uv_1}(\mu_1) + 1]$ in $O(1)$ time. This shows that SNODE$(P(\nu))$ can be executed in $O(|V_\nu| + |E_\nu| + \Delta\alpha_\nu)$ time.

Therefore the entire algorithm STARSHAPEDRAW can be implemented to run in $O(|V| + |E|)$ time. This proves Theorem 4. □

5. Concluding remarks

In this paper, we introduced a new notion of drawing, called star-shaped drawing for biconnected planes graphs, as an extension of convex drawing for internally triconnected plane graphs. Using the new concepts of configuration and fringe corners, we present a linear-time algorithm for constructing a star-shaped drawing of a biconnected plane graph with a given set A of concave corners.

As a follow-up of this paper, we studied the problem of finding a star-shaped drawing with the *minimum number of concave corners* for a given biconnected plane graph, and the problem of finding the *best plane embedding* of biconnected planar graphs which gives the minimum number of concave corners [12–14].

- We characterize a necessary and sufficient condition for a subset B of prescribed corners to admit a star-shaped drawing D whose concave corners are contained in B [13], as an extension of Thomassen's classical characterization of biconnected plane graphs with a prescribed boundary that have convex drawings [23].
- Given a non-negative cost for each corner in G, we prove that a star-shaped drawing with the minimum cost can be found in linear time, where the cost of a drawing is defined by the sum of costs of concave corners in the drawing [14].
- We present a linear-time algorithm for finding the best plane embedding of a biconnected *planar* graph G, i.e., an embedding that gives the minimum number of concave corners, based on the effective use of lower bounds [12].

We conclude with some open problems in this research area:

- *Characterization*: In this paper, we present a sufficient condition for a set A of corners in a plane graph G to admit a star-shaped drawing whose concave corners are exactly given by the corners in A. However, a necessary and sufficient condition for a set A of corners in a plane graph G to admit such a star-shaped drawing is still open.
- *Star-shaped grid drawing*: An interesting variation of star-shaped drawings is to study star-shaped grid drawing with small area. Note that the corresponding problem of convex grid drawings is well studied by a number of authors [1,3,4,18].
- *Approximation algorithm*: The problem of minimizing the number of concave faces is NP-hard [16]. It would be interesting to design an approximation algorithm for the problem.

Acknowledgements

This research was partially supported by a Scientific Grant in Aid from the Ministry of Education, Science, Sports and Culture of Japan.

References

[1] N. Bonichon, S. Felsner, M. Mosbah, Convex drawings of 3-connected plane graphs, in: Proc. of the 12th International Symposium on Graph Drawing (GD 2004), in: Lecture Notes in Computer Science, vol. 3383, Springer, 2005, pp. 60–70.
[2] N. Chiba, T. Yamanouchi, T. Nishizeki, Linear algorithms for convex drawings of planar graphs, in: Progress in Graph Theory, Academic Press, 1984, pp. 153–173.
[3] M. Chrobak, M.T. Goodrich, R. Tamassia, Convex drawings of graphs in two and three dimensions, in: Proc. of the 12th Annual Symposium on Computational Geometry (SoCG 1996), ACM Press, 1996, pp. 319–328.
[4] M. Chrobak, G. Kant, Convex grid drawings of 3-connected planar graphs, International Journal of Computational Geometry and Applications 7 (1997) 211–223.
[5] G. Di Battista, P. Eades, R. Tamassia, I.G. Tollis, Graph Drawing: Algorithms for the Visualization of Graphs, Prentice-Hall, 1999.
[6] G. Di Battista, R. Tamassia, On-line planarity testing, SIAM J. on Comput. 25 (5) (1996) 956–997.
[7] I. Fáry, On straight line representations of planar graphs, Acta Sci. Math. Szeged 11 (1948) 229–233.
[8] C. Gutwenger, P. Mutzel, A linear time implementation of SPQR-trees, in: Proc. of the 8th International Symposium on Graph Drawing (GD 2000), in: Lecture Notes in Computer Science, vol. 1984, Springer, 2001, pp. 77–90.
[9] S.-H. Hong, H. Nagamochi, Convex drawings of graphs with non-convex boundary, in: Proc. of the 32nd International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2006), in: Lecture Notes in Computer Science, vol. 4271, Springer, 2006, pp. 113–124.
[10] S.-H. Hong, H. Nagamochi, Convex drawings of graphs with non-convex boundary constraints, Discrete Applied Mathematics 156 (12) (2008) 2368–2380.
[11] S.-H. Hong, H. Nagamochi, Convex drawings of hierarchical planar graphs and clustered planar graphs, Journal of Discrete Algorithms, in press.
[12] S.-H. Hong, H. Nagamochi, Star-shaped drawings of planar graphs, in: Proc. of 18th International Workshop on Combinatorial Algorithms (IWOCA 2007), College Publications, 2008, pp. 78–92.
[13] S.-H. Hong, H. Nagamochi, Star-shaped drawings of graphs with fixed embedding and concave corner constraints, in: Proc. of the 14th International Computing and Combinatorics Conference (COCOON 2008), in: Lecture Notes in Computer Science, vol. 5092, Springer, 2008, pp. 405–414.
[14] S.-H. Hong, H. Nagamochi, Star-shaped drawings of plane graphs with cost function on concave corners, submitted for publication.
[15] J.E. Hopcroft, R.E. Tarjan, Dividing a graph into triconnected components, SIAM J. on Comput. 2 (1973) 135–158.
[16] G. Kant, Algorithms for drawing planar graphs, Ph.D. Dissertation, Department of Computer Science, University of Utrecht, Holland, 1993.
[17] M. Kaufmann, D. Wagner (Eds.), Drawing Graphs: Methods and Models, Lecture Notes in Computer Science Tutorial, vol. 2025, Springer, 2001.
[18] K. Miura, S. Nakano, T. Nishizeki, Convex grid drawings of four-connected plane graphs, International Journal of Foundations of Computer Science 17 (5) (2006) 1031–1060.
[19] K. Miura, M. Azuma, T. Nishizeki, Convex drawings of plane graphs of minimum outer apices, International Journal of Foundations of Computer Science 17 (5) (2006) 1115–1128.
[20] T. Nishizeki, M.S. Rahman, Planar Graph Drawing, World Scientific, 2004.
[21] K. Sugiyama, Graph Drawing and Applications, World Scientific, 2002.
[22] K.S. Stein, Convex maps, Proc. Amer. Math. Soc. 2 (1951) 464–466.
[23] C. Thomassen, Plane representations of graphs, in: Progress in Graph Theory, Academic Press, 1984, pp. 43–69.
[24] W.T. Tutte, Convex representations of graphs, Proc. of London Math. Soc. 10 (3) (1960) 304–320.
[25] W.T. Tutte, Graph Theory, Encyclopedia of Mathematics and Its Applications, vol. 21, Addison-Wesley, Reading, MA, 1984.
[26] K. Wagner, Bemerkungen zum Vierfarbenproblem, Jahresbericht Deutsch. Math. Verein. 46 (1936) 26–32.