Angle and Distance Constraints on Tree Drawings

Ulrik Brandes and Barbara Schlieper*

Department of Computer & Information Science, University of Konstanz

Abstract. We consider planar drawings of trees that must satisfy constraints on the angles between edges incident to a common vertex and on the distances between adjacent vertices. These requirements arise naturally in many applications such as drawing phylogenetic trees or route maps. For straight-line drawings, either class of constraints is always realizable, whereas their combination is not in general. We show that straight-line realizability can be tested in linear time, and give an algorithm that produces drawing satisfying both groups of constraints together in a model where edges are represented as polylines with at most two bends per edge or as continuously differentiable curves.

1 Angle and Distance Constraints

We are interested in planar drawings of simple undirected graphs $G = (V, E)$. Throughout this paper, we assume that G is planar and let $n = |V|$ denote the number of vertices and $m = |E|$ the number of edges. We are particularly interested in drawing *trees* $T = (V, E)$, i. e. graphs that are connected and acyclic. Denote by $T(e, v)$ the tree obtained from splitting T by removing $e \in E$ and choosing the component that contains $v \in V$. For any $r \in V$ let T_r be the tree *rooted* at r, and $T_r(v)$ the subtree of all descendants of v (including v itself). Clearly, $T_r(r) = T$ and $T_r(v) = T(v, e)$ if $v \neq r$ and e is the unique first edge on the path from v to r. With $\{v, w\}$ we refer to the undirected edge incident to v and w.

The implications of the following two types of constraints on drawings of a graph G are investigated.

Distance constraints: A drawing of a graph $G = (V, E)$ satisfies *distance constraints* $\delta : E \to \mathbb{R}^+$, if all pairs of adjacent vertices $\{v, w\} \in E$ are exactly at distance $\delta(v, w)$.

Angle constraints: A graph is said to be *embedded* (combinatorially), if the, say, counterclockwise cyclic ordering of edges incident to the same vertex is prescribed. For an embedded graph $G = (V, E)$, let $A \subseteq E \times E$ be the *angle set*, where $(e_1, e_2) \in A$ iff both edges share a vertex v and e_2 is the counterclockwise next edge after e_1 around v.

* To whom correspondence should be directed: schliepe@inf.uni-konstanz.de.
This author gratefully acknowledges financial support from the state of Baden-Württemberg (Landesgraduiertenförderungsgesetz scholarship).

M. Kaufmann and D. Wagner (Eds.): GD 2006, LNCS 4372, pp. 54–65, 2007.

A drawing of an embedded graph $G = (V, E)$ satisfies *angle constraints* $\alpha : A \to (0, 2\pi]$, if the angle between all pairs $(e_1, e_2) \in A$ is exactly $\alpha(e_1, e_2)$. Note that α is frequently called an *angle assignment*.
A necessary requirement for angle constraints to be satisfiable is that they sum to 2π around every vertex, and to $(d_G(f) - 2)\pi$ around every inner face f with $d_G(f)$ vertices. Such a set of angle constraints is called *locally consistent*, and we assume that all given angle constraints are.

A graph with angle and/or distance constraints is called *realizable in the straight-line model* (or *straight-line realizable* for short), if there exists a planar straight-line drawing in the plane, such that all constraints are satisfied.

2 Straight-Line Realizability

Testing straight-line realizability is known to be $\mathcal{NP}$-complete for both distance-constrained graphs (even if all edges are constrained to have unit length) [5] and angle-constrained graphs [7]. For trees, arbitrary distance and angle constraints can be satisfied, though not necessarily in the same drawing.

Theorem 1. *For any tree $T = (V, E)$ with locally consistent angle constraints α (or distance constraints δ), a planar straight-line drawing satisfying α (or δ) can be determined in linear time.*

Proof. A drawing of T satisfying any locally consistent angle constraints, can be determined in linear time using a simple postorder traversal to create a balloon layout [9]: starting with an empty circle of arbitrary radius around each leaf, parent edges are stretched such that the enclosing circles of subtrees rooted at siblings do not intersect when satisfying the angle constraints.

An algorithm for drawing any distance-constrained tree in linear time is given in [1]. □

Note that straight-line drawings of trees with both angle and distance constraints are completely determined (up to translation and rotation).

Theorem 2. *Straight-line realizability of trees with both angle and distance constraints can be tested in linear time.*

Proof. We show that straight-line realizability testing is equivalent to testing simplicity of polygonal chains, which can be done in linear time [3].

Since a polygonal chain can be viewed as a tree with angle and distance constraints, trees cannot be tested faster than polygonal chains. On the other hand, an embedded tree with l leaves can be covered by l paths $p_1, \ldots, p_l$ connecting each leaf with the first leaf encountered in a right-first search. Due to the given constraints, each path corresponds to a polygonal chain, and the tree is realizable if and only if all p_i, $1 \leq i \leq l$ are simple. Since the union of these paths corresponds to an Euler tour around the tree, the total size of the polygonal chains is $2m = 2n - 2$. □

3 Polyline Representation

If an angle and distance constrained tree is not straight-line realizable, it can still be drawn without edge intersections by allowing polylines. In the remainder of the section, we will prove the following theorem.

Theorem 3. *For a tree, a planar polyline drawing that satisfies locally consistent angle and distance constraints and has at most two bends per edges can be determined in linear time.*

In the first step we calculate an initial layout of T_r for an arbitrary root $r \in V$ with the length-preserving algorithm of [1] and given edge lengths δ. We exploit an invariant characteristic of the layouts computed.

Theorem 4. *For a tree with given vertex positions a planar polyline drawing that satisfies locally consistent angle constraints and has at most two bends per edge can be determined in linear time, if the given vertex positions are such that disjoint subtrees are contained in disjoint wedges.*

This is a special case of the problem to find a drawing of a planar graph with fixed vertices and pre-specified angles between the edges incident to the same vertex [2]. A related problem is embedding a planar graph on a fixed set of points in the plane. The graph can be drawn without edge intersections using at most two bends per edge in polynomial time, if the mapping between the vertices V and the points P is not fixed [8]. However, if the mapping is fixed i. e., each vertex has a fixed position such that the straight-line drawing is not necessarily planar, up to $O(n)$ bends per edge can be needed to guarantee planarity and this bound is known to be asymptotically optimal in the worst case [10]. Note that these strategies do not yield drawings that satisfy angle constraints. Angle constraints have to be satisfied for example when drawing graphs with good angular resolution. A planar graph can be embedded and drawn planar with at most one bend per edge, such that for each $(e_1, e_2) \in A$ sharing a vertex $v \in V$ it is $\alpha(e_1, e_2) \geq \frac{1}{d(v)}$ where $d(v)$ denotes the degree of v [4]. For not necessarily planar graphs angular resolution and the number of edge crossings can be improved modifying a force-directed graph drawing algorithm into an algorithm for drawing graphs with curved edges [6]. Note that for these drawings no distances constraints are to be satisfied.

In our drawings edges will be represented as polylines. The polyline of an edge $\{v, w\}$ will be determined by the endpoints of two control segments incident to v and w.

We guarantee the planarity in two steps: For a vertex $v \in V$

- we determine the direction of each initial control segment.
- we determine the length for each initial control segment.

In Sect. 3.1 we define a rotation angle β to guarantee certain situations regarding the angles of the initial control segments incident to a vertex v and those of the straight lines from v to the corresponding neighbors. In Sect. 3.2

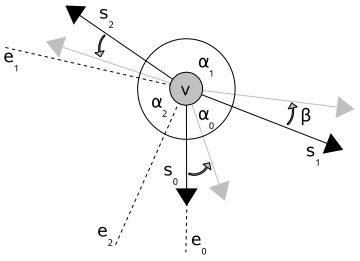


Fig. 1. Rotating the angle template

we determine the control segment lengths to avoid intersections in the remaining situations.

We will focus on each vertex a constant number of times and look at all incident edges so the overall running time is linear.

Let v be a vertex we focus on with k incident control segments $s_0 \dots s_{k-1}$ (in counterclockwise order) belonging to the k polylines of edges $e_0 \dots e_{k-1}$ to the k neighbors $w_0 \dots w_{k-1}$ of v. We refer to the absolute angle of a control segment s_i with $\gamma_i = \gamma_0 + \sum_{t=0}^{k-1} \alpha_t$ and l_i its length. Further, let s'_i be the control segment for e_i incident to w_i with angle γ'_i and length l'_i. Let p_i denote the target point of s_i and p'_i the target point of s'_i, λ_i the angle of the line from v to w_i and λ'_i the angle of the same line, but from w_i to v.

3.1 Control Segment Angles

Since only the relative angles between consecutive control segments incident to v are given, to determine the final layout we have to choose the absolute angle for one of the control segments. We start with $\gamma_0 = \lambda_0$ and then rotate the whole angle template by an angle β i. e., update each angle $\gamma_i = \gamma_i + \beta$. For a neighbor w_i of v the *angular deviation* is $\theta(i) = (\gamma_i - \lambda_i) \bmod 2\pi$. We say that w_j *lies between* s_i and w_i, if $(\gamma_i - \lambda_j) \bmod 2\pi < (\gamma_i - \lambda_i) \bmod 2\pi$.

We assume that for each neighbor w_i the endpoint p'_i of the control segment incident to w_i lies on the same side of the line (v, p_i) and will assure this when determining the control segment lengths (see (7),(8)). The following two situations will cause intersections in the starting configuration, because e_i will be intersecting with e_0 (see Fig. 2 for illustration). For a pair s_i, w_i we say that

$$\begin{aligned} &i \text{ and } 0 \text{ are } \textit{clockwise crossing} \text{ if} \\ &\theta(i) \leq \pi \text{ and } w_0, s_0 \text{ lie between } s_i \text{ and } w_i, \end{aligned} \tag{1}$$

$$\begin{aligned} &i \text{ and } 0 \text{ are } \textit{counterclockwise crossing} \text{ if} \\ &\theta(i) \geq \pi \text{ and } w_0, s_0 \text{ lie between } w_i \text{ and } s_i. \end{aligned} \tag{2}$$

(a) clockwise crossing (b) counterclockwise crossing

Fig. 2. Two cases in the start situation

Lemma 1. *At any vertex $v \in V$, there are either clockwise or counterclockwise crossings, if any.*

Proof. Assume i_1 fulfills Equation (1) and i_2 Equation (2), then s_0 lies between s_{i_1} and s_{i_2} and w_0 lies between w_{i_2} and w_{i_1}. The orders of the control segments and of the neighbors are fixed, hence $i_1 < i_2$ and $i_1 > i_2$, which is a contradiction. □

If a vertex v has a neighbor w_i such that i and 0 are clockwise crossing we rotate v's angle template counterclockwise. If i and 0 are counterclockwise crossing for an $0 \leq i \leq k-1$ this is the mirrored case and can be solved by rotating clockwise. After rotating the angle template we will have (see Fig. 3):

Lemma 2 (counterclockwise sheering). *For $0 \leq i, j \leq k-1$, w_j lies between w_i and s_i with angular deviation $\theta(i) > \pi$, if and only if also s_i lies between w_j and s_j with $\theta(j) > \pi$.*

Lemma 3 (clockwise sheering). *For $0 \leq i, j \leq k-1$, w_j lies between s_i and w_i with angular deviation $\theta(i) < \pi$, if and only if also s_i lies between s_j and w_j with $\theta(j) < \pi$.*

In these two situations we can avoid intersections by determining the control segment lengths.

We say s_j is *pulled between* s_i and w_i, if it was $(\lambda_i - \gamma_j) \bmod 2\pi < \pi$ before rotation, but is $(\lambda_i - \gamma_j) \bmod 2\pi > \pi$ and s_j lies between s_i and w_i after rotation. We say s_i is *pushed over* w_j, if it was $(\lambda_j - \gamma_i) \bmod 2\pi < \pi$ before rotation, but is $(\lambda_j - \gamma_i) \bmod 2\pi > \pi$ and w_j lies between s_i and w_i after rotation.

Of all pairs s_i, w_i for which i and 0 are clockwise crossing let $i_{\min}$ be the one for which the angular deviation $\theta(i)$ is minimal and of all pairs s_i, w_i for which i and 0 are not clockwise crossing but $\theta(i) < \pi$ let $i_{\max}$ be the one for which $\theta(i)$ is maximal.

We rotate the angle template counterclockwise by an angle β with

$$0 \leq \pi - \theta(i_{\min}) < \beta < \pi - \theta(i_{\max}) \leq \pi \tag{3}$$

Lemma 4. *For every vertex v there is a feasible rotation β, i. e. $\theta(i_{\min}) > \theta(i_{\max})$.*

(a) counterclockwise sheering (b) clockwise sheering

Fig. 3. Two solvable situations

Proof. For $i_{\max} > i_{\min}$ the vertex $w_{i_{\max}}$ lies between w_0 and $w_{i_{\min}}$ because the order of neighbors is fixed. $i_{\max}$ and 0 are not clockwise crossing, hence $s_{i_{\max}}$ lies between w_0 and $w_{i_{\max}}$. So it must be $\theta(i_{\min}) > \theta(i_{\max})$ because both $s_{i_{\max}}$ and $w_{i_{\max}}$ lie between $s_{i_{\min}}$ and $w_{i_{\min}}$ with $\theta(i_{\max}) < \pi$. The proof for the case $i_{\max} > i_{\min}$ is analogous. □

Lemma 5. *These bounds are sharp.*

Proof. For $\beta \leq \pi - \theta(i_{\min})$ it would still be $\theta(i_{\min}) \leq \pi$ after rotation and both w_0 and s_0 would still lie between $s_{i_{\min}}$ and $w_{i_{\min}}$, hence $e_{i_{\min}}$ would be intersecting with e_0.

For $\beta \geq \pi - \theta(i_{\max})$ it would be $\theta(i_{\max}) \geq \pi$ after rotation and both $w_{i_{\min}}$ and $s_{i_{\min}}$ would lie between $w_{i_{\max}}$ and $s_{i_{\max}}$, hence $e_{i_{\min}}$ would be intersecting with $e_{i_{\max}}$. □

Within these bounds we can now optimize any objective function, for example the sum over all squared angular deviations $\theta(i)^2$ for $0 \leq i \leq k-1$. See [2] for other reasonable objective functions.

To proof Lemma 2 and Lemma 3 we first proof the following:

Lemma 6. *If $\theta(i) > \pi$ after rotation for $0 \leq i \leq k-1$, before and after rotation neither w_0 nor s_0 can lie between w_i and s_i.*

Proof. If $\theta(i) > \pi$ before rotation, w_0 and s_0 cannot have lain between w_i and s_i, because i and 0 would have been counterclockwise crossing. We rotate counterclockwise by an angle $\beta < \pi$, hence they cannot after rotation as well.

If $\theta(i) \leq \pi$ before rotation but $\theta(i) > \pi$ after, i and 0 must have been clockwise crossing before rotation, so w_0 and s_0 cannot lie between w_i and s_i after rotation. □

Lemma 7. *If $\theta(i) < \pi$ after rotation for $0 \leq i \leq k-1$, not both w_0 and s_0 can lie between s_i and w_i.*

Proof. Before rotation i and 0 cannot have been clockwise crossing, so by rotating counterclockwise by an angle $\beta < \pi$ from the start situation $\lambda_0 = \gamma_0$, it can only happen that either s_i is pushed over w_0 or s_0 is pulled between s_i and w_i. □

Corollary 1. *For the angular deviation of a vertex w_i, $\theta(i) = \pi$ is impossible for any $0 \leq i \leq k-1$.*

Proof. of Lemma 2 (Lemma 3 can be proven with the same ideas)

"⇒" Neither s_0 nor w_0 can lie between w_i and s_i (Lemma 6) so w_0 cannot lie between w_i and w_j and we have $j < i$. Because the order of control segments is fixed s_0 cannot lie between s_i and s_j, hence s_0 must lie between s_j and w_i and s_0 lies between s_j and w_j. If it was $\theta(j) \leq \pi$ s_0 must have been pulled between s_j and w_j (Lemma 7) so s_0 must have lain between w_i and s_i which is a contradiction to Lemma 6.

"⇐" Neither s_0 nor w_0 can lie between w_j and s_j (Lemma 6) so s_0 cannot lie between s_i and s_j and we have $j < i$. Because the order of neighbors is fixed w_0 cannot lie between w_i and w_j, hence w_0 must lie between s_j and w_i and w_0 lies between s_i and w_i. If it was $\theta(i) \leq \pi$ s_i must have pushed over w_0 (Lemma 7), so also s_j must have pushed over w_0, hence w_0 must have lain between w_j and s_j which is a contradiction to Lemma 6.

□

3.2 Control Segment Lengths

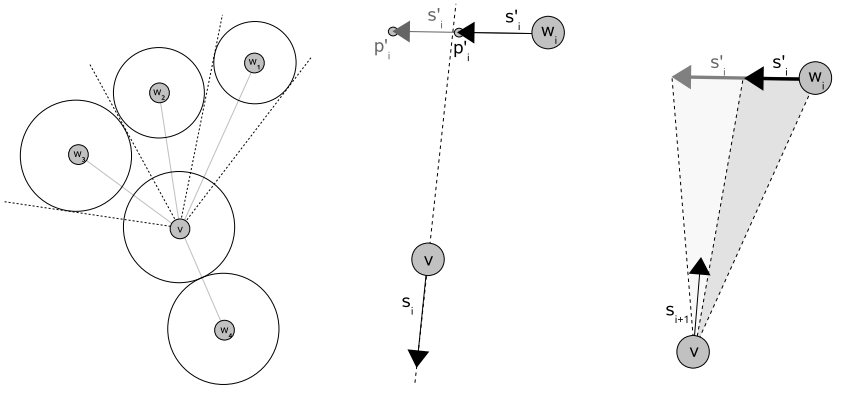
In the remaining situations we can avoid intersections by determining first the radius r_v of the circle containing all control segments incident to a vertex v and then the lengths l and l' of an edge e's control segments. We have to make sure that none of the three segments of an edge e's polyline can intersect with any segment of another polyline. The maximal possible radius r_v is determined such that:

- control segments incident to different neighbors of v can not intersect
- the control segments incident to v can not intersect with a control segment incident to any of v's neighbors

For an edge $e = \{v, w\}$ the maximal possible length of e's control segment s incident to v is determined such that:

- neither the middle segment of e nor the control segment s can intersect with a control segment incident to another neighbor
- neither the middle segment of e nor the control segment s can intersect with the middle segment of another edge

When we computed the initial layout of T_r a wedge with size ω_v was assigned to each vertex v in which v's subtree $T(v)$ was lying and that was divided among the children. The children's wedges are rooted in v. We now look at the unrooted tree T and all the wedges of v's neighbors are rooted in v. The vertex w_0, that had been v's parent in T_r, is now lying in an opposed wedge with size $\omega_{w_0} = 2\pi - \omega_v$



(a) wedges for vertex v (b) line (v, p_i) (c) range of w_i

Fig. 4.

(see Fig. 4(a)). To guarantee that the control segments incident to one neighbor are not intersecting with the control segments of another neighbor, we determine for $0 \leq i \leq k-1$:

$$r_{w_i} < \begin{cases} \sin \frac{\omega_{w_i}}{2} \cdot \parallel w_i - v \parallel_2 & \text{if } \omega_{w_i} < \pi \\ \sin(\pi - \frac{\omega_{w_i}}{2}) \cdot \parallel w_i - v \parallel_2 & \text{otherwise} \end{cases} \tag{4}$$

We also have to guarantee that the control segments incident to v are not intersecting with the control segments incident to one of v's neighbors:

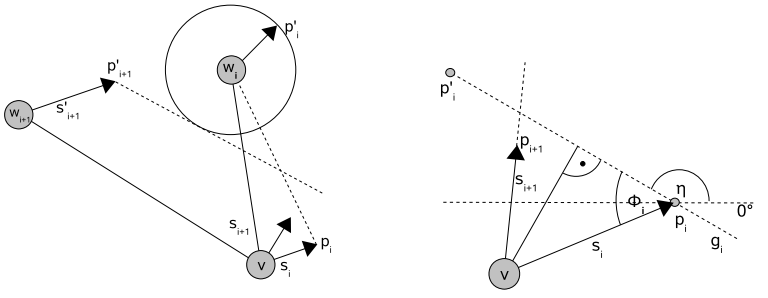
$$r_v \leq \frac{1}{2} \min\{\delta\{e_i\}\}_{0 \leq i \leq k-1} \tag{5}$$

The computations in (4) and (5) will be done first for each vertex $v \in V$ to determine r_v, which we will need in the following.

A neighbor w_i cannot be involved both in a clockwise and a counterclockwise sheering. Let i and $(i+1) \bmod k$ (we will write $i+1$ in the remainder) be counterclockwise sheering. We have to further determine the length l_{i+1} of the control segment s_{i+1}.

The polyline of an edge e_{i+1} might intersect with e_i or another edge incident to w_i if a line through p'_{i+1} and p_{i+1} would be intersecting with the circle containing all control segments incident to w_i. We can avoid this by choosing l_{i+1} such that s_{i+1} is not intersecting with a tangent line to w_i's circle through p'_{i+1}. Further, s_{i+1} must not intersect with the line $g_i = (p_i, p'_i)$. We use the maximal length r_{w_i} here to determine possible coordinates of p'_i. If we focus on w_i later, s'_i might have to be shortened, hence s_{i+1} must not intersect with the line (p_i, w_i) and the tangent line to w_i's circle through w_{i+1}. See Figure 5(a) for illustration.

The computations for a length l_{i+1} such that s_{i+1} is not intersecting with one of these lines, are all very similar. We show the computation here for the line



(a) Lines (b) length of s_{i+1}

Fig. 5. Bounding control segments

$g_i = (p_i, p'_i)$ (note that the length l_i has to be fixed already) with absolute angle η_i and ϕ_i the angle between g_i and s_i in (6) (see Figure 5(b) for illustration):

$$l_{i+1} < \frac{\sin\phi_i \cdot l_i}{\sin(\pi - \phi_i - \gamma_{i+1} + \gamma_i)} \tag{6}$$

In Sect. 3.1 we assumed that the neighbor w_i and the endpoint p'_i of the control segment incident to w_i lie on the same side of the line (v, p_i). We assure this by choosing l'_i, the length of s'_i, such that s'_i does not intersect with the line (v, p_i) (see Fig. 4(b)):

If $(\gamma'_i - \lambda'_i) \bmod 2\pi > \pi$ and $\theta(i) > \pi$:

$$l'_i < \frac{\sin(\gamma_i + \pi - \lambda_i) \cdot \delta(e_i)}{\sin(-\gamma_i + \lambda_i - \lambda'_i + \gamma'_i)} \tag{7}$$

If $(\gamma'_i - \lambda'_i) \bmod 2\pi < \pi$ and $\theta < \pi$:

$$l'_i < \frac{\sin(\lambda_i - \gamma_i - \pi) \cdot \delta(e_i)}{\sin(-\lambda_i + \gamma_i - \gamma'_i + \lambda'_i)} \tag{8}$$

Let w_i be one neighbor of v. We call the (smallest) sector of w_i's wedge, in which the control segment s'_i incident to w_i is lying, the *range* of w_i. If another control segment s_j incident to v is lying within this range, the polylines of e_i and e_j can be intersecting without i and j clockwise or counterclockwise sheering. We avoid this by choosing the length l'_i of s'_i such that s'_i is not intersecting with the line (v, p_j) (see Fig. 4(c)), the computation is analog to (7) and (8).

With these control segments lengths we can draw the edges' polylines without intersections (see also Lemma 9). In Fig. 7(a) a tree after determining the vertex positions in displayed with an arbitrary rotation angle for each vertex v and control segment lengths smaller than r_v. In Fig. 7(b) the rotation angle is determined like shown in Sect. 3.1 and the control segment lengths like in Sect. 3.2.

4 Curve Representation

Instead of using polylines we can also represent the edges as smooth curves. A cubic Bézier curve is determined by two endpoints b_0, b_3 and two inner control points b_1, b_2. We call the segments $\overline{b_0 b_1}$ and $\overline{b_3 b_2}$ the (initial) control segments, while $\overline{b_1 b_2}$ is called inner segment. A Bézier curve is contained in the convex hull of its defining points, and the tangents at its endpoints are collinear with the initial control segments, so the outgoing angle of a Bézier curve is the angle of its control segment. For the Bézier curve of an edge we use the control segments of the corresponding polyline as initial control segments. We refer to the Bézier curve from v to w_i for $0 \leq i \leq k-1$ with c_i and H_i the convex hull of its control points. In the remainder of this section we will proof the following theorem.

Theorem 5. *For a tree, a planar drawing that satisfies locally consistent angle and distance constraints while representing edges as continuously differentiable curves consisting of at most two cubic Bézier curves and a straight line segment can be determined in linear time.*

Even if the polylines of two edges are not intersecting, the hulls of the corresponding Bézier curves can intersect. Three borders of a curve's hull are the line segments of the corresponding polyline, and when determining the control segment lengths we avoided most intersections, but in case of a clockwise or counterclockwise sheering we have to split an edge's curve by adding control segments.

Let i and $i+1$ be counterclockwise sheering. When all the control segment lengths in the tree are determined, we split a curve c_i. Let q_i be the intersection point of the lines (v, p_{i+1}) and g_i. The splitting point m_i must lie on g_i between the point q_i and p_i. Rooted at m_i we add two diametral opposed control segments s_i^{m1} and s_i^{m2} on g_i, with s_i^{m1} pointing to p_i and s_i^{m2} pointing to p_i'. (See Fig. 6 for illustration.) Now we can describe c_i by two Bézier curves smoothly attached (continuously differentiable, because the angle at m_i has size π) one by s_i and s_i^{m1} and one by s_i' and s_i^{m2}. When we later focus on w_i, we might have to split c_i to avoid intersections with curves incident to w_i. If then the splitting point cannot lie between q_i and p_i, we will add a second splitting point on g_i, otherwise one splitting point will be sufficient.

This procedure induces that l_i has to be calculated first. Therefore we create a vertex list L^+ sorted by increasing index and $\theta(i) < \pi$ for each $w_i \in L^+$.

If for v there is a pair i and $(i-1) \bmod k$ clockwise sheering this is symmetrically the same situation and will be solved with the same strategy. We will need a vertex list L^- sorted by decreasing index and $\theta(i) > \pi$ for each $w_i \in L^-$. We can create both types of vertex lists testing all neighbors w_i for $0 \leq i \leq k-1$ in counterclockwise order.

Lemma 8. *Neither the vertex list L^+ nor L^- will contain all neighbors of v.*

Proof. We start with the angular deviation $\theta(0) = 0$. If we do not have to rotate v's angle template the neighbor w_0 will be in none of the vertex lists. Iff

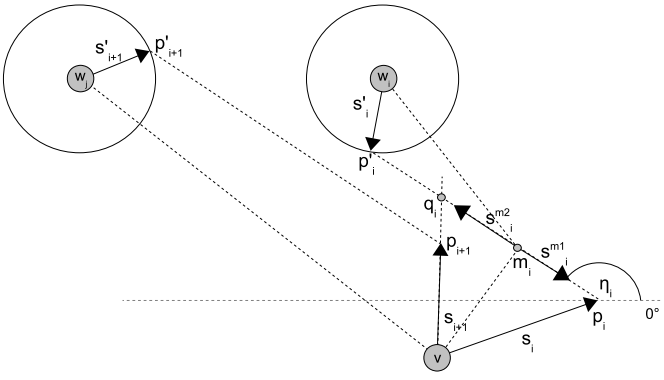


Fig. 6. Splitting a curve

there is a neighbor w_i of v such that i and 0 are clockwise crossing we rotate counterclockwise by an angle $\beta < \pi$ until $\theta(i) > \pi$. After rotation it will be $\theta(0) < \pi$, hence w_0 and w_i will be in different vertex lists. Rotating clockwise is the mirrored case. □

Lemma 9. *The resulting tree layout is planar.*

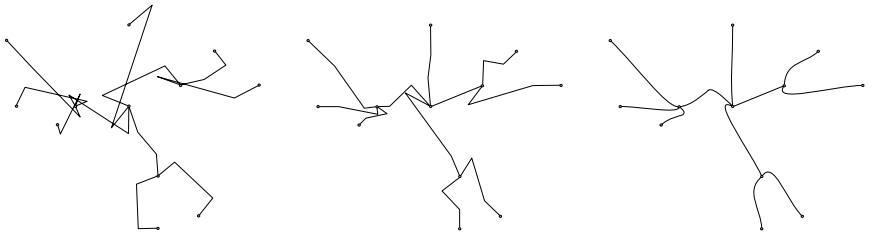
Proof. For any vertex v the curve or polyline of an edge e_{i+1} incident to v can cause problems, if the hull H_{i+1} is intersecting with the wedge of another neighbor of v. With the strategy presented previously we made sure, that H_i and H_{i+1} are not intersecting.

By determining the length of s_{i+1} in (6) we also made it impossible for H_{i+1} to intersect with any wedge or control segment of one of w_i's neighbors different from v. With this, H_{i+1} can not intersect with any line from a point in w_i's circle to a point in the wedge of one of w_i's neighbors different from v, thus H_j cannot intersect with any hull of a curve incident to w_i or any vertex in the subtree $T(w_i, \{w_i, v\})$.

Hull H_{i+1} intersecting with the wedge of the neighbor w_{i+2} is the mirrored case. □

5 Discussion

We presented efficient algorithms for drawing trees with constraints on distances between adjacent vertices and angles between incident edges. There is plenty of opportunity for further work. For instance, we would like to address angle and distance constraints together to improve both vertex placements and angle rotations, and enlarge the class of graphs on which our methods work. A major challenge is to implement the rotation method of [2] so that planarity is maintained always.



(a) random rotation (b) polylines (c) curves

Fig. 7. Drawings of a tree with fixed vertices and angles

References

[1] C. Bachmaier, U. Brandes, and B. Schlieper. Drawing phylogenetic trees. In *Proc. 16th Intl. Symp. Algorithms and Computation (ISAAC '05)*, volume 3827 of *LNCS*, pages 1110–1121. Springer, 2005.

[2] U. Brandes, G. Shubina, and R. Tamassia. Improving angular resolution in visualizations of geographic networks. In *Data Visualization: Proc. 2nd Joint EUROGRAPHICS and IEEE TCVG Symp. Visualization, VisSym*, pages 23–33. Springer, 29–30 2000.

[3] B. Chazelle. Triangulating a simple polygon in linear time. *Discrete Comput. Geom.*, 6(5):485–524, 1991.

[4] C. C. Cheng, C. A. Duncan, M. T. Goodrich, and S. G. Kobourov. Drawing planar graphs with circular arcs. In *Proc. Graph Drawing 1999*, volume 1731 of *LNCS*, pages 117–126. Springer, 1999.

[5] P. Eades and N. C. Wormald. Fixed edge-length graph drawing is $\mathcal{NP}$-hard. *Discrete Applied Mathematics*, 28:111–134, 1990.

[6] B. Finkel and R. Tamassia. Curvilinear Graph Drawing Using the Force-Directed Method. In *Proc. Graph Drawing 2004*, volume 3383 of *LNCS*, pages 448–453. Springer, 2004.

[7] A. Garg. On drawing angle graphs. In *Proc. Graph Drawing 1994*, volume 894 of *LNCS*, pages 84–95. Springer, 1994.

[8] M. Kaufmann and R. Wiese. Embedding vertices at points: Few bends suffice for planar graphs. *Journal of Graph Algorithms and Applications*, 6(1):115–129, 2002.

[9] G. Melançon and I. Hermann. Circular drawing of rooted trees. Technical report 9817, Reports of the Center for Mathematics and Computer Science, 1998.

[10] J. Pach and R. Wenger. Embedding planar graphs at fixed vertex locations. In *Proc. Graph Drawing 1998*, volume 1547 of *LNCS*, pages 263–274. Springer, 1998.