A Radial Adaptation of the Sugiyama Framework for Visualizing Hierarchical Information

Christian Bachmaier

Abstract— In radial drawings of hierarchical graphs the vertices are placed on concentric circles rather than on horizontal lines and the edges are drawn as outwards monotone segments of spirals rather than straight lines as it is both done in the standard Sugiyama framework. This drawing style is well suited for the visualisation of centrality in social networks and similar concepts. Radial drawings also allow a more flexible edge routing than horizontal drawings, as edges can be routed around the center in two directions. In experimental results this reduces the number of crossings by approximately 30 percent on average. Few crossings are one of the major criteria for human readability.

This paper is a detailed description of a complete framework for visualizing hierarchical information in a new radial fashion. Particularly, we briefly cover extensions of the level assignment step to benefit by the increasing perimeters of the circles, present three heuristics for crossing reduction in radial level drawings, and also show how to visualize the results.

Index Terms— graph drawing, radial, crossing reduction, Sugiyama framework, spiral segments

I. INTRODUCTION

HIERARCHICAL graph layout is the method of choice for visualizing a general direction of flow, e. g., data or information, in relational data. Thereby vertices are usually drawn on parallel horizontal lines, and edges are drawn as y-monotone polylines that may bend when they intersect a level line. The standard drawing algorithm [1] for visualizing flow in vertical direction consists of four phases: *cycle removal* (reverses appropriate edges to eliminate cycles), *level assignment* (assigns vertices to levels and introduces dummy vertices to represent edge bends), *crossing*

reduction (permutes vertices on the levels), and *coordinate assignment* (assigns x-coordinates to vertices, y-coordinates are implicit through the levels). See [2] for an extended overview.

The novelty of this paper is that we draw the level lines as concentric circles instead of as parallel horizontal lines and thus visualize flow from the center outwards. Further, in analogy to y-monotone straight line edges used in horizontal level drawings, we draw the edges as segments of spirals, unless they are radially aligned, in which case they are drawn as straight lines. This results in strictly monotone curves from inner to outer levels and ensures that the segments do not cross inner level lines or each other unnecessarily. The apparent advantage of radial level drawings is that level graphs can be drawn with fewer edge crossings. It is also more likely that a graph can be drawn without any crossings at all, since the set of level planar graphs is a proper subset of the set of radial level planar graphs [3]. Note that radial level drawings are different from circular drawings [4]–[6] where only one circle contains all vertices and do not comply with radial drawings [7] where edges are drawn straight line and level lines are not equidistant. Further, in contrast to both, here "inner level edges" with both end vertices on a common level are prohibited. See Fig. 1 for an example.

Radial drawings are not new, for example see the hierarchical graph drawings of [8], the ring diagrams of [9], or the radial tree drawings of [10], which are common for free trees. Radial level drawings are also common, e. g., in the study of social networks [11], [12]. There vertices model *actors* and edges represent *relations* between the actors. The importance (*centrality*) of a vertex is expressed by its distance (*closeness*) to the center, i. e., a position on a low level. Radial level drawings are also well suited for level graphs with an increasing number of vertices on higher levels. For example, in a graph that shows which Web pages are reachable from a given start page by following k hyperlinks, higher levels are likely to contain many vertices while there are only few vertices on the lower
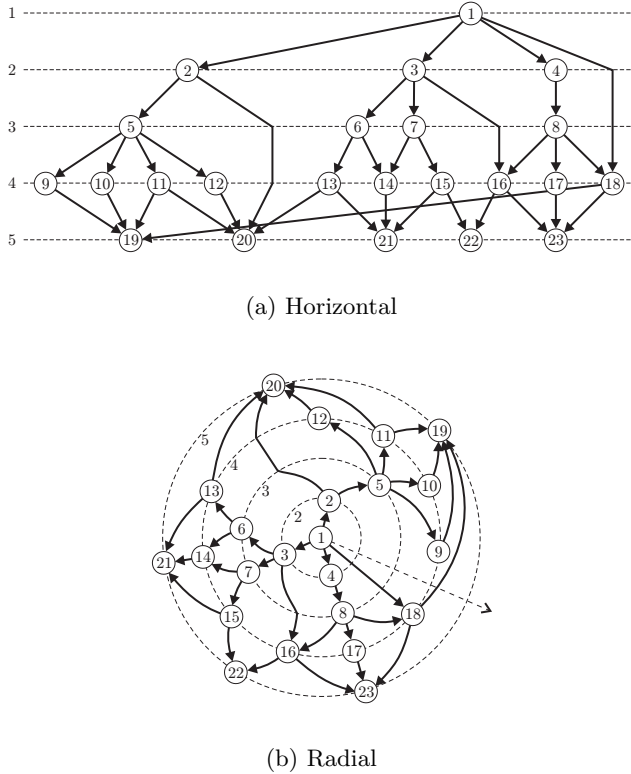
(a) Horizontal

(b) Radial

Fig. 1. Drawings of a level graph

levels. Other potential applications may be found in [13], [14].

The paper is organized as follows: After some preliminary definitions in the next section, we present a complete framework in Sugiyama style to create radial level drawings of hierarchical graphs. This is done by introducing methods for radial level assignment in Sect. III, radial crossing reduction in Sect. IV, and radial coordinate assignment [15] in Sect. V. We omit the cycle removal step since it does not differ from the horizontal case and standard algorithms can be used. See [2] for an overview on that topic. We conclude in Sect. VI.

II. PRELIMINARIES

A k-*level graph* $G = (V, E, \phi)$ is a directed acyclic graph (DAG) with a level assignment $\phi\colon V \to \{1, 2, \ldots, k\}$, which partitions the vertex set into $k \leq |V|$ pairwise disjoint subsets $V = V_1 \dot\cup V_2 \dot\cup \cdots \dot\cup V_k$, $V_i = \phi^{-1}(i)$, $1 \leq i \leq k$, such that $\phi(u) < \phi(v)$ for each edge $(u, v) \in E$. Particularly, $k = 1$ implies that $E = \emptyset$. An edge (u, v) is *short* if $\phi(v) - \phi(u) = 1$; otherwise it is *long* and *spans* the levels $\phi(u)+1, \ldots, \phi(v)-1$. A level graph without long edges is *proper*. Any level graph can be made proper by subdividing each long edge $(u, v) \in E$ by the introduction of new *dummy vertices*

$v_i \in B_i$, $i = \phi(u) + 1, \ldots, \phi(v) - 1$, $\phi(v_i) = i$. We draw dummy vertices as small black circles, Fig. 6(a), or edge bends, Fig. 1. The edges of a proper level graph are also called (edge) *segments*. If both end vertices of a segment are dummy vertices, it is called an *inner segment*. Let $N = |V \cup B| + |B| + |E|$ denote the size of the proper level graph $G = (V \cup B, E, \phi)$ where V contains the original vertices and $B = B_1 \dot\cup B_2 \dot\cup \ldots \dot\cup B_k$ contains the dummy vertices with $|B| \leq k|E| \leq |V||E|$.

An *ordering* of a proper level graph is a partial order $\prec$ of $V \cup B$ such that $u \prec v$ or $v \prec u$ iff $\phi(u) = \phi(v)$. This is equivalent to a definition of the vertex *positions* on level i as a bijective function $\pi_i\colon V_i \cup B_i \to \{0, \ldots, |V_i| + |B_i| - 1\}$ with $u \prec v \Leftrightarrow \pi_i(u) < \pi_i(v)$ for any two vertices $u, v \in V_i \cup B_i$. Let $\pi = (\pi_i)_{1 \leq i \leq k}$. We call an ordering of a level graph a *horizontal embedding*. In case that an ordering admits a drawing without edge crossings (except common end points), it is called a *level planar embedding*. Throughout the paper let $N^-(v) = \{\, u \mid (u, v) \in E \,\}$ denote the predecessors of $v \in V$. A vertex is called a *source (sink)*, if there is no incoming (outgoing) incident edge. Let $\mathrm{sgn}\colon \mathbb{R} \to \{-1, 0, 1\}$ denote the signum function.

III. RADIAL LEVEL ASSIGNMENT

THE basic problem is the same as in horizontal level drawings: In this phase a given DAG is to be transformed into a level graph by assigning the vertices to levels. Thus, any existing level assignment algorithm for horizontal level drawings can directly be used for radial level drawings. The optimization criteria, however, slightly change: Radial level drawings use k concentric circles to place the vertices of the k levels. Contrary to the constant line lengths in horizontal level drawings, the perimeters of the circles grow longer with ascending level numbers: On an outer circle, there is space for more vertices than on an inner circle. Thus, we investigate how level assignment methods can be extended to take advantage of this.

A straight-forward idea is to apply the longest path level assignment method from outer to inner levels: First, each sink of the graph is assigned to the highest level. For the remaining vertices the level is recursively defined by $\phi(v) = \min\{\, \phi(w) \mid (v, w) \in E \,\} - 1$. This puts each vertex on the outermost possible level while minimizing the number of levels k. There is no explicit balancing of level sizes, however.

For a better vertex distribution, an extension of the Coffman/Graham algorithm [16] can be used that explicitly takes into account the growing perimeter of

the circles. Coffman and Graham compute a leveling where the number of vertices per level is bounded by a given constant W. We change this bound to be a function $w(i)$ which grows proportionally with the index i of a level: $w(i) = W \cdot i$. The first phase of the algorithm remains unchanged, but we apply it using the opposite edge direction: After removing transitive edges in linear time, an appropriate numbering $o\colon V \to \{0, \ldots, |V| - 1\}$ of the vertices is computed: Initially, all vertices are unnumbered. We consecutively choose one vertex at a time and assign the next ascending number to it. The vertex is chosen so that it has no unnumbered successors and that the numbers of the successors are minimal regarding a specific ordering of integer sets: a set of vertex numbers is considered less than another one, if the maximum is less. If the maximum of both sets is equal, the next smaller value is compared, and so on. In the second phase the algorithm places one vertex at a time, starting with the vertex numbered with $|V| - 1$ on level $i = 1$ and filling the levels from inner to outer circles. In one step it places the next unleveled vertex $v \in V$ with maximum $o(v)$ whose predecessors are already leveled. If level i is full, i.e., if i contains $w(i)$ vertices, or if v has a predecessor u with $\phi(u) = i$, then a new level is started, i.e., i is increased by 1. The level of v is then set to $\phi(v) = i$.

As the last step of the level assignment phase the level graph is made proper, because for drawing level graphs it is necessary to know where long edges should be routed, i.e., between which two vertices on a spanned level. Thus all long edges are subdivided in proper segments by new dummy vertices B in $\mathcal{O}(k|E|)$ running time. In the following, we will only consider proper level graphs.

IV. RADIAL CROSSING REDUCTION

REGARDLESS of whether the leveling of a level graph is given by the application or if it has been computed by one of the algorithms in the previous section, the next step towards a hierarchical drawing is to compute an embedding. In horizontal hierarchical drawings the embedding is fully defined by the vertex ordering π. For radial embeddings it is also necessary to know where the (w.l.o.g. counter-clockwise) orderings start and end on each level. Therefore, we introduce a *ray* that tags this borderline between the vertices, cf. Fig. 1(b). The ray is a straight halfline from the center to infinity between the vertices on each level with extremal positions. Edges crossed by the ray are called *cut edges*. In horizontal drawings of level graphs a crossing between two edges only depends

(a) Edge $(1, 3)$ drawn counter-clockwise and clockwise (dotted)

(b) $\psi\left((1, 2)\right) = +3$

Fig. 2. Offsets of edges

on the orderings of the end vertices. In radial level drawings, however, it is also necessary to consider the direction in which the edges are wound around the center, clockwise or counter-clockwise. Furthermore, edges can also be wound around the center multiple times. We call this the offset $\psi\colon E \to \mathbb{Z}$ of an edge. Thereby, $|\psi(e)|$ counts the crossings of an edge $e \in E$ with the ray. If $\psi(e) < 0$ ($\psi(e) > 0$), e is a *clockwise* (*counter-clockwise*) cut edge, i.e., the sign of $\psi(e)$ reflects the mathematical direction of rotation, see Fig. 2. If $\psi(e) = 0$, then e is not a cut edge and thus needs no direction information. Observe that a cut edge cannot cross the ray clockwise and counter-clockwise simultaneously. We define a *radial embedding* $\mathcal{E}$ of a graph $G = (V, E, \phi)$[1] to consist of the vertex ordering π and of the edge offsets ψ, i.e., $\mathcal{E} = (\pi, \psi)$.

Compared to horizontal drawings there is an additional freedom in radial drawings without changing the crossing number: *rotation* of a level i. A clockwise rotation moves the vertex v with minimum position on the ordered level $\phi(v) = i$ over the ray by setting $\pi_i(v)$ to the maximum on i. The other values of π_i are updated accordingly. For an illustration see Fig. 3, where $v = 5$. A counter-clockwise rotation is defined symmetrically. Rotations do not modify the "cyclic order", i.e., the neighborhood of every vertex on its radial level line is preserved. However, the offsets of the edges incident to v must be updated. If rotating clockwise, the offsets of incoming edges of v are reduced by 1 and the offsets of outgoing edges are increased by 1. The offset updates for rotating counter-clockwise are symmetric. Depending on the implementation, rotation needs $\mathcal{O}(\deg(v))$ resp. $\mathcal{O}(|V| + \deg(v))$ running time.

The most common technique for crossing reduction in proper level graphs is to only consider two consecu-

---

[1]We need not to distinguish between original and dummy vertices in this phase. Thus here V denotes both of them for an easy notation.

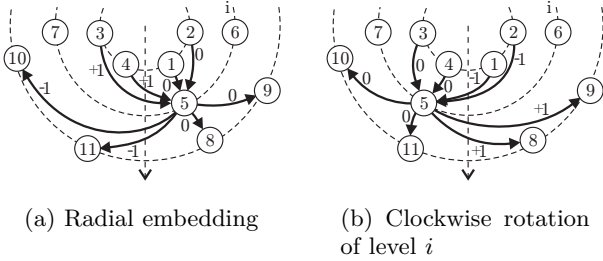(a) Radial embedding   (b) Clockwise rotation of level i

Fig. 3. Rotation

tive levels at a time in multiple top-down and bottom-up passes. Starting with an arbitrary permutation of the first level, subsequently the ordering of one level is fixed, while the next one is reordered to minimize the number of crossings in-between. This *one-sided two-level crossing reduction problem* is $\mathcal{NP}$-hard [17] and well-studied. For radial level embeddings we follow the same strategy and consider the *radial one-sided two-level crossing reduction problem*. Given a 2-level graph $G = (V_1 \dot\cup V_2, E, \phi)$ and an ordering π_1 of the first level, our objective is to compute an ordering of the second level and offsets for the edges with few crossings.

A. Properties

Crossings between edges in radial embeddings depend on their offsets and on the order of the end vertices. There can be more than one crossing between two edges, if they have very different offsets. We denote the number of crossings between two edges $e_1, e_2 \in E$ in an embedding $\mathcal{E}$ by $\chi_\mathcal{E}(e_1, e_2)$. Intuitively, this number is approximately equal to the difference of the offsets $|\psi(e_2) - \psi(e_1)|$. The exact formula is slightly different, however, with a small shift depending on the vertex ordering, see Lemma 1.[2] The (radial) crossing numbers of a radial embedding $\mathcal{E}$ and of a level graph $G = (V, E, \phi)$ are then naturally defined as $\chi(\mathcal{E}) = \sum_{\{e_1, e_2\} \subseteq E, e_1 \neq e_2} \chi_\mathcal{E}(e_1, e_2)$ and $\chi(G) = \min\{\chi(\mathcal{E}) \mid \mathcal{E} \text{ is a radial embedding of } G\}$.

Lemma 1: Let $\mathcal{E} = (\pi, \psi)$ be a radial embedding of a 2-level graph $G = (V_1 \dot\cup V_2, E, \phi)$. Then the number of crossings between two edges $e_1 = (u_1, v_1) \in E$ and $e_2 = (u_2, v_2) \in E$ is

$$\chi_\mathcal{E}(e_1, e_2) =$$
$$\max\left\{0, \left|\psi(e_2) - \psi(e_1) + \frac{b-a}{2}\right| + \frac{|a|+|b|}{2} - 1\right\},$$

[2] Although high offsets are never useful for a low number of crossings, we nevertheless provide the general result, not only to show that it also can be computed in constant time, but also since it is an interesting problem in itself.

TABLE I
THE CROSSING NUMBER IN RELATION TO $\delta = \psi(e_2) - \psi(e_1)$

u_1 vs. u_2	v_1 vs. v_2	$\psi(e_2)$ vs. $\psi(e_1)$	$\chi_\mathcal{E}(e_1, e_2)$	u_1 vs. u_2	v_1 vs. v_2	$\psi(e_2)$ vs. $\psi(e_1)$	$\chi_\mathcal{E}(e_1, e_2)$
≺	≺	<	$\|\delta\|$	≻	≻	>	$\|\delta\|$
≺	≺	=	0	≻	≻	=	0
≺	≺	>	$\|\delta\|$	≻	≻	<	$\|\delta\|$
≺	=	<	$\left|\delta - \frac{1}{2}\right| - \frac{1}{2}$	≻	=	>	$\left|\delta + \frac{1}{2}\right| - \frac{1}{2}$
≺	=	=	0	≻	=	=	0
≺	=	>	$\left|\delta - \frac{1}{2}\right| - \frac{1}{2}$	≻	=	<	$\left|\delta + \frac{1}{2}\right| - \frac{1}{2}$
≺	≻	<	$\|\delta - 1\|$	≻	≺	>	$\|\delta\| + 1$
≺	≻	=	1	≻	≺	=	1
≺	≻	>	$\|\delta - 1\|$	≻	≺	<	$\|\delta\| + 1$
=	≺	<	$\left|\delta - \frac{1}{2}\right| - \frac{1}{2}$	=	≻	>	$\left|\delta - \frac{1}{2}\right| - \frac{1}{2}$
=	≺	=	0	=	≻	=	0
=	≺	>	$\left|\delta - \frac{1}{2}\right| - \frac{1}{2}$	=	≻	<	$\left|\delta - \frac{1}{2}\right| - \frac{1}{2}$
=	=	<	$\|\delta\| - 1$	=	=	>	$\|\delta\| - 1$
=	=	=	0				

where $a = \operatorname{sgn}(\pi_1(u_2) - \pi_1(u_1))$ and
$$b = \operatorname{sgn}(\pi_2(v_2) - \pi_2(v_1)) \ .$$

Proof: In analogy to horizontal embeddings, edge crossings do not depend on the exact position of the end vertices, but only on the relative ordering ($\prec$, $\succ$, or $=$) and on the edge offsets. We can assume w. l. o. g. that $\psi(e_1) = 0$, because in any embedding we can rotate the whole second level multiple times without changing π_2 or the offset difference $\delta = \psi(e_2) - \psi(e_1)$. This leads to $3 \cdot 3 \cdot 3 = 27$ cases, which are straightforward to prove, see Tab. IV-A. ∎

Corollary 1: Let $\mathcal{E}$ be a radial embedding of a 2-level graph $G = (V_1 \dot\cup V_2, E, \phi)$. Swapping the position of two vertices $v, w \in V_2$ changes the number of crossing between two edges $(\cdot, v), (\cdot, w) \in E$ by at most 1.

Before we show our radial crossing reduction algorithms, we first discuss some more properties which follow from radial level lines.

Lemma 2: Let $G = (V_1 \dot\cup V_2, E, \phi)$ be a 2-level graph and let $e_1 = (u_1, v) \in E$ and $e_2 = (u_2, v) \in E$ be two edges with a common target vertex v. Then in any crossing minimal radial embedding $\mathcal{E} = (\pi, \psi)$ of G, $\pi_1(u_1) < \pi_1(u_2)$ implies $\psi(e_2) - \psi(e_1) \in \{0, 1\}$.

Proof: Assume that $\psi(e_2) - \psi(e_1) \notin \{0, 1\}$. Then Lemma 1 implies $\chi_\mathcal{E}(e_1, e_2) > 0$. We choose an arbitrary crossing between e_1 and e_2 and show how the embedding can be modified to reduce the number of crossings, see Fig. 4(a) for an illustration. We exchange the routing of e_1 and e_2 between v and the crossing: e_1 is routed along the old course of e_2 until it reaches the crossing. The routing from there to u_1 is not changed. We symmetrically do the same
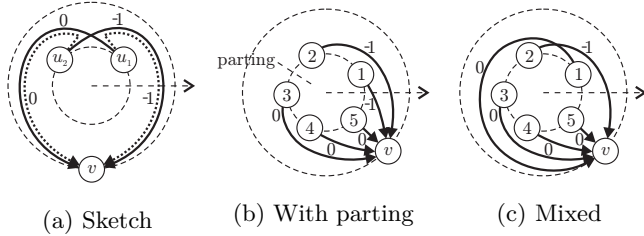
(a) Sketch   (b) With parting   (c) Mixed

Fig. 4.   Not all offset combinations for edges $(\cdot, v) \in E$ result in few crossings

with e_2. In the new embedding e_1 and e_2 have one crossing less and the number of crossings has not changed otherwise, contradicting the assumption and proving the lemma. ∎

Because of this result, it is clear that only embeddings need to be considered, where there is a clear *parting* between all edges incident to the same vertex as in Fig. 4(b). The parting is that position of the edge list of v that separates the two subsequences with offsets ψ_0 resp. $\psi_0 + 1$. Otherwise unnecessary crossings are generated between the incident edges, see Fig. 4(c). We also only consider radial embeddings with small edge offsets, because large offsets correspond to very long edges which are difficult to follow. Obviously, winding edges more than once around the center can only increase its number of crossings. Thus, only the offsets -1, 0, and 1 are used in our algorithms.

Lemma 3: Radial one-sided two-level crossing minimization is $\mathcal{NP}$-hard.

Proof: We show the $\mathcal{NP}$-hardness by reduction from the horizontal one-sided two-level crossing minimization problem, which is known to be $\mathcal{NP}$-hard [17]. Given a 2-level graph $G = (V_1 \dot{\cup} V_2, E, \phi)$ with a fixed permutation π_1 of the first level, we construct a new 2-level graph $G' = (V_1' \dot{\cup} V_2', E', \phi')$ by adding a barrier as follows: G is extended by $|E|^2$ new vertices $x_0, \dots, x_{|E|^2-1}$ at the end of the first level $\pi_1'(x_i) = |V_1| + i$, $\phi'(x_i) = 1$ and a new vertex y on the second level $\phi'(y) = 2$ that is connected to them by new edges $e_0, \dots, e_{|E|^2-1}$, $e_i = (x_i, y)$.

Let $\mathcal{E}' = ((\pi_1', \pi_2'), \psi)$ be a radial embedding of G' that has a minimum number of crossings subject to π_1'. We can assume w. l. o. g. (because of rotation and Lemma 2) that $\pi_2'(y) = |V_2|$ and $\psi(e_i) = 0$ for all new edges. Then none of the new edges has a crossing with any of the original edges, because this would lead to $|E|^2$ crossings, contradicting the minimality of the embedding. Thus, there are no cut edges, and $\pi_2 = \pi_2'|_{V_2}$ is a solution of the original horizontal one-sided

two-level crossing minimization problem. ∎

As a consequence, we use heuristics for an efficient solution of the problem. In the following, we present three different approaches, extending some well-known [2] horizontal one-sided two-level crossing reduction methods, namely the barycenter, median, and sifting heuristics.

B. Cartesian Barycenter Heuristic

In the horizontal barycenter crossing reduction method for every vertex in V_2 the average value of the positions of its neighbors in V_1 is computed. Afterward V_2 is sorted according to this values following the rule of thumb "shorter edges have less crossings than longer edges". With some restrictions, this method can be directly used to compute a radial embedding: The horizontal vertex ordering defines a radial vertex ordering, and all edge offsets are set to 0. This neglects the additional freedom of radial edge routing, however, and therefore introduces more crossings than necessary. The result is especially bad for vertices whose neighbors on the first level are far apart. If, for an extreme example, a vertex is only adjacent to the first and last vertex of the first level, its best position is obviously near the ray, labeling one of the edges as a cut edge. But the horizontal algorithm cannot do that, and therefore produces an out-of-balance embedding. Even worse, the result depends on the current position of the ray.

One approach to improve that could be to rotate the first level before computing the barycenter values to an appropriate position, or maybe even use different rotations for different vertices. We propose a simpler, yet equally promising method. The basic idea stays the same: each vertex should be close to the average position of its neighbors. However, we use the terms "average" and "position" in a geometric sense. We assume the vertices of the first level V_1 to be uniformly distributed on a unit circle, according to the given ordering π_1. This defines Cartesian coordinates $(x(u), y(u)) \in \mathbb{R}^2$ for each $u \in V_1$. Then we compute for each $v \in V_2$ the *Cartesian barycenter*[3]

$$\text{bary}(v) = \left(\frac{\sum_{u \in N^-(v)} x(u)}{|N^-(v)|}, \frac{\sum_{u \in N^-(v)} y(u)}{|N^-(v)|} \right)$$

of its predecessors $N^-(v)$ and sort the vertices circularly around the origin, i. e., by the angles of $\text{bary}(v)$

---

[3]Note that the division by $|N^-(v)|$ can be omitted in an implementation, because it does not change the polar angle of $\text{bary}(v)$.

in polar coordinates,

$$\beta(v) = \arctan \frac{y\left(\mathrm{bary}(v)\right)}{x\left(\mathrm{bary}(v)\right)}$$
$$+ \pi \cdot H\left(-x\left(\mathrm{bary}(v)\right)\right) \cdot \mathrm{sgn}\left(y\left(\mathrm{bary}(v)\right)\right),$$

where $H(x) = 0$ for $x \leq 0$ and $H(x) = 1$ for $x > 0$ is the unit step function. Many programming languages provide a specialized function `atan2(x, y)` for this purpose.

After sorting, we distribute the vertices of the second level uniformly on a concentric circle with radius 2 and choose for the offset of each edge one of -1, 0, or $+1$, whichever leads to the shortest edge in a geometric sense. Obviously, this algorithm has the same running time as its horizontal version:

Theorem 1: The running time of the Cartesian barycenter heuristic is $\mathcal{O}(|E| + |V| \log |V|)$.

C. Cartesian Median Heuristic

The Cartesian median heuristic is similar to the Cartesian barycenter heuristic. The only difference is that we take component-wise the x and y median instead of the component-wise barycenter. The running time stays the same, since $\mathrm{med}(v)$ can be computed in $\mathcal{O}(N^-(v))$, see [18]. The median values depend on the underlying coordinate system (origin and rotation). But since we use the same coordinates for all median computations, this is no problem. Rotated coordinate systems, however, might lead to different results.

D. Radial Sifting Heuristic

As a contrast to the fast and simple algorithms described above, we also developed an extension of the sifting heuristic, which is slower but generates fewer crossings. Sifting was originally introduced as a heuristic for vertex minimization in ordered binary decision diagrams [19] and later adapted for the (horizontal) one-sided crossing minimization problem [20]. The idea is to keep track of the objective function while moving a vertex $v \in V_2$ along a fixed ordering of all other vertices in V_2. Then v is placed to its locally optimal position. The method is thus an extension of the greedy-switch heuristic [21], where neighbors are only swapped if this does not increase the number of crossings. For crossing reduction the objective function is the number of crossings between the edges incident to the vertex under consideration and all other edges. Consecutively testing each vertex $v \in V_2$ on each position once is called a *sifting round.*

The efficient computation of crossing numbers in sifting for horizontal embeddings is based on the *crossing matrix.* Its entries correspond to the number of crossings caused by pairs of vertices in a particular relative ordering and can be computed as a preprocessing step. Whenever a vertex is placed to a new position, only a small number of updates is necessary. For radial embeddings, however, the crossings matrix cannot be computed in advance, because two vertices cannot be said to be in a particular (linear) relative order on radial levels.

Let $\mathcal{E} = (\pi, \psi)$ and $\mathcal{E}' = (\pi', \psi)$ be two embeddings of G, where $\mathcal{E}'$ is computed from $\mathcal{E}$ by swapping the vertex $v \in V_2$ and its successor $w \in V_2$ according to π_2, i.e., $\pi_2'(w) = \pi_2(v)$ and $\pi_2'(v) = \pi_2(v) + 1$. Since swapping positions of v and w only affects crossings of incident edges, the number of crossings in $\mathcal{E}'$ is efficiently computed as

$$\chi(\mathcal{E}') = \chi(\mathcal{E}) - c_{\mathcal{E}}(v, w) + c_{\mathcal{E}'}(v, w), \text{ where}$$
$$c_{\mathcal{E}}(v, w) = \sum_{u \in N^-(v)} \sum_{x \in N^-(w)} \chi_{\mathcal{E}}\left((u, v), (x, w)\right) .$$

Unfortunately, we cannot directly transfer the ideas of [4] for the efficient computation of that formula, because in radial sifting the crossing numbers also depend on the edge offsets, which are not constant in our approach. A change in the offset of an edge may affect all other edges. Therefore, the overall running time of this part of the algorithm for one sifting round is $\mathcal{O}(|E|^2)$ instead of $\mathcal{O}(|V||E|)$. The total running time of the algorithm, however, is dominated by the next step, anyway.

In addition to the position of vertex v, we also have to compute the offsets of the incident edges. As v moves along the second level circle in counterclockwise direction, we update the offsets accordingly. Because of Lemma 2 we do not consider each possible offset combination for each position of v. Intuitively, the parting of the edges should move around the first level circle in the same direction as v, but on the opposite side of the circle. Otherwise, the edges incident to v get longer and tend to increase the number of crossings. Thus, we only decrease edge offsets by 1, starting with $\psi(e) = 1$ for all incident edges e, and we also do this one by one in the order of the end vertices on level 1. The decision for which offsets are updated at which position of v is made subject to whether this leads to an improvement or not. Note that the parting may move around level 1 twice, as offsets are decreased from 1 to -1.

Algorithm 1 shows one round of radial sifting. We do not try the position $|V_2|-1$, because it is equivalent to position 0 modulo rotation.

Algorithm 1: RADIAL-SIFTING

Input: Two level graph $G = (V_1 \dot\cup V_2, E, \phi)$ with radial embedding $\mathcal{E} = (\pi, \psi)$

Output: Updated embedding $\mathcal{E}$, i.e., positions π_2 and offsets ψ

1 **foreach** $v \in V_2$ with $\deg(v) > 0$ **do**

2   // put v to first position
3   **foreach** $w \in V_2$ with $\pi_2(w) < \pi_2(v)$ **do**
4          $\pi_2(w) \leftarrow \pi_2(w) + 1$
5      $\pi_2(v) \leftarrow 0$
6   let $\{v = v_0, \ldots, v_{|V_2|-1}\} \leftarrow V_2$ be ordered by π_2
7   let $E_v \leftarrow \{e_0, \ldots, e_{\deg(v)-1}\}$ be the edges $(u, v) \in E$ ordered by $\pi_1(u)$

8   // init offsets as 1
9   **foreach** $e_v \in E_v$ **do** $\psi(e_v) \leftarrow 1$

10   // init counters for pos, offset, parting, #cross.
11     $i^* \leftarrow 0$; $j^* \leftarrow j \leftarrow 0$; $l^* \leftarrow l \leftarrow 0$; $c^* \leftarrow c \leftarrow 0$

12   // search best position for v
13   **for** $i \leftarrow 0$ **to** $|V_2| - 2$ **do**
14   **repeat**
15   // try to improve part. by red. next offset
16             $c_1 \leftarrow \sum_{e \in E} \chi_{\mathcal{E}}(e_l, e)$
17             $\psi(e_l) \leftarrow j$
18             $c_2 \leftarrow \sum_{e \in E} \chi_{\mathcal{E}}(e_l, e)$
19   // if successful, try again, else restore
20   **if** $c_2 \leq c_1$ **then**
21                 $c \leftarrow c - c_1 + c_2$
22                 $l \leftarrow l + 1$
23   **if** $l = \deg(v)$ **then**
24                     $j \leftarrow j - 1$
25                     $l \leftarrow 0$
26   **else** $\psi(e_l) \leftarrow j + 1$
27   **until** $c_1 < c_2$

28   // store best pos, offset, parting, #cross.
29   **if** $c < c^*$ **then** $i^* \leftarrow i$; $j^* \leftarrow j$; $l^* \leftarrow l$; $c^* \leftarrow c$

30   // swap v and v_{i+1} and update #cross.
31   let $E_{v_{i+1}}$ be the set of edges $(\cdot, v_{i+1}) \in E$ incident to v_{i+1}
32         $c \leftarrow c - \sum_{e_v \in E_v} \sum_{e_{v_{i+1}} \in E_{v_{i+1}}} \chi_{\mathcal{E}}(e_v, e_{v_{i+1}})$
33         $\pi_2(v_{i+1}) \leftarrow i$; $\pi_2(v) \leftarrow i + 1$
34         $c \leftarrow c + \sum_{e_v \in E_v} \sum_{e_{v_{i+1}} \in E_{v_{i+1}}} \chi_{\mathcal{E}}(e_v, e_{v_{i+1}})$

35   // place v to best position
36   **foreach** $w \in V_2$ with $\pi_2(w) \geq i^*$ **do**
37          $\pi_2(w) \leftarrow \pi_2(w) + 1$
38      $\pi_2(v) \leftarrow i^*$
39   // set best offsets for v's incident edges
40   **for** $i \leftarrow 0$ **to** $l^* - 1$ **do** $\psi(e_i) \leftarrow j^*$
41   **for** $i \leftarrow l^*$ **to** $\deg(v) - 1$ **do** $\psi(e_i) \leftarrow j^* + 1$

Theorem 2: Given a 2-level graph $G = (V, E, \phi)$, the algorithm RADIAL-SIFTING runs in $\mathcal{O}(|V|^2 \cdot |E|)$ time.

Proof: For each node $v \in V_2$ the content of the repeat-until loop in lines 14–27 is executed $\mathcal{O}(|V| + \deg(v))$ times: once per position, and additionally once per shifted parting. It is thus executed $\mathcal{O}(|V|^2 + |E|)$ times in total. As the running times of lines 16 and 18 are $\mathcal{O}(|E|)$, the repeat-until loop contributes $\mathcal{O}(|V|^2 \cdot |E| + |E|^2)$ to the overall running time.

The only other relevant part are lines 32 and 34, which are executed once for each pair (v, v_{i+1}). Since the summation needs $\mathcal{O}(\deg(v) \cdot \deg(v_{i+1}))$, the total running time of this part is $\mathcal{O}(|E|^2)$ and is therefore dominated by the above. ∎

To allow a harmonic drawing of the computed embedding in the next phase a final postprocessing which rotates level 2 with respect to uniform edge lengths is useful, e.g., see Fig. 5. Since our algorithm starts with an offset of 1 for every edge and stops at the first best parting among several others which are as good, a straightforward drawing of the embedding is twisted too much (in counter-clockwise direction). Thus the sum of the absolute edge lengths can be reduced by rotating level 2 with Algorithm 2: While assuming to have a drawing with uniform vertex distribution on both levels, we compute the average angle spanned by the edges and rotate the whole level 2 by this amount. However, this $\mathcal{O}(|E|)$ time postprocessing is only for aesthetic reasons and does neither affect the number of crossings nor the asymptotic running time.

E. Experimental Results

To analyse the performance of our heuristics, we have implemented them in Java. Further, we have realized the corresponding horizontal versions to compare the resulting number of crossings with the radial algorithms. We have tested the implementations using a total number of 5000 random graphs: 50 graphs for each combination of the following parameters: $|V_1| = |V_2| \in \{20, 40, 60, 80, 100\}$ and $|E|/|V_2| \in \{1, \ldots, 20\}$.

The experimental results in the Appendix show that all radial heuristics generate fewer crossings than their horizontal equivalents, experimentally by a factor of 0.7. This is a very encouraging result, since the running times of the radial algorithms (except sifting) are similar, see Fig. 10. Like in the horizontal case [22], Cartesian barycenter on average leaves slightly fewer crossings than Cartesian median. Another similarity is that radial sifting is the best among all three radial heuristics, but also the slowest. Usually only few
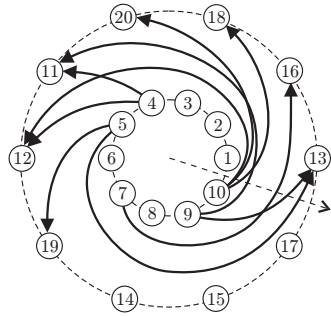
(a) Twisted embedding
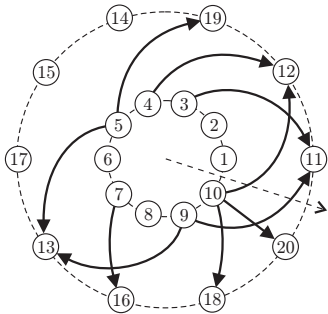
(b) Figure 5(a) clockwise rotated by 4

Fig. 5.　Postprocessing to reduce absolute edge lengths

Algorithm 2: RADIAL-SIFTING-POSTPROCESS

Input: Radial embedding $\mathcal{E} = (\pi, \psi)$ of a two level graph $G = (V_1 \dot\cup V_2, E, \phi)$

Output: Updated embedding $\mathcal{E}$, i.e., positions π_2 and offsets ψ

1 *// init vertex distances and average angle*
2 $\epsilon_1 \leftarrow \frac{2\pi}{|V_1|}$; $\epsilon_2 \leftarrow \frac{2\pi}{|V_2|}$; $\delta \leftarrow 0$

3 *// compute average angle spanned by edges*
4 **foreach** $e = (u, v) \in E$ **do**
5 $\quad$ $\alpha_u \leftarrow \pi_1(u) \cdot \epsilon_1$; $\alpha_v \leftarrow \pi_2(v) \cdot \epsilon_2$
6 $\quad$ $\delta \leftarrow \delta + (\alpha_u - \alpha_v) + 2\pi \cdot \psi(e)$

7 $\delta \leftarrow \frac{\delta}{|E|}$

8 *// if necessary, rotate level 2*
9 $r \leftarrow \lfloor \frac{\delta}{\epsilon_2} + \frac{1}{2} \rfloor$
10 **if** $r < 0$ **then**
11 $\quad$ **for** $i \leftarrow 1$ **to** $|r|$ **do**
12 $\quad\quad$ Counter-clockwise rotate level 2

13 **else if** $r > 0$ **then**
14 $\quad$ **for** $i \leftarrow 1$ **to** r **do**
15 $\quad\quad$ Clockwise rotate level 2

---

sifting rounds ($3 - 5$ for reasonable problem instances) are necessary to reach a local optimum for all vertices simultaneously, and the largest reduction of crossings usually occurs in the first round. In our experiments we further observed that the quality of radial sifting does not depend much on the quality of the initial embedding. However, a poor initialization increases the number of sifting rounds needed and thus raises the absolute running time.

V. Radial Coordinate Assignment

AS already mentioned, in radial level drawings we draw the edge segments as segments of a spiral, unless they are radially aligned, in which case they are drawn as straight lines. This results in strictly monotone curves from inner to outer levels and ensures that segments do not cross inner level lines or unnecessarily each other. This phase is usually constrained not to change the vertex orderings computed previously, what is especially useful if the input embedding is a planar embedding, e.g., as generated by [3]. Further, the drawing algorithm should support commonly accepted criteria for readability and aesthetics, like small area, good separation of (dummy) vertices within a level, length and slope of edges, straightness of long edges, and balancing of edges incident to the same vertex. In our opinion edge bends in radial level drawings tend to be even more disturbing than in horizontal level drawings. Thus we base our algorithm on the approach of Brandes/Köpf [23] which guarantees at most two bends per edge. Further it prioritizes vertical alignment, which helps us to obtain radial alignment. The criterion of small area in horizontal coordinate assignment, i.e., to obtain small width, turns to uniform distribution of the vertices on the radial levels. As a consequence, a user parameter δ like in Sect. V-A is not needed. Since the input embedding for this phase maintains the position $\pi(v)$ for every vertex $v \in V \cup B$, the position of the ray is implicitly evident, i.e., on each level it lies between the two vertices with extremal positions.

A. Horizontal Coordinate Assignment

There are several algorithms for horizontal coordinate assignment [1], [24]–[31] using different approaches for the optimization of various objective functions or iterative improvement techniques. Most interesting is the Brandes/Köpf algorithm [23], which generates at most two bends per edge and draws every inner segment vertically if no two inner segments cross. Further it minimizes the horizontal stretch of

segments and also gives good results for the other aesthetic criteria. The algorithm has $\mathcal{O}(N)$ running time and is fast in practice. For level planar embeddings Eades et al. [32] presented an algorithm that does not generate bends at all. However, it may need exponential area.

Since the horizontal drawing algorithm of Brandes/Köpf [23] is the basis of our radial drawing algorithm, we give an extended overview. It consists of three basic steps: vertical alignment, horizontal compaction, and balancing. The first two steps are carried out four times. After that, the four results are combined in the balancing step.

1) Vertical Alignment: The objective is to consecutively align each vertex with its left upper, right upper, left lower, and right lower median neighbor. Here and in Sect. V-A.2 we describe the alignment to the left upper median, the other three passes are analogous.

At the beginning all segments are removed from the graph which do not lead to an upper median neighbor, i. e., only candidates for vertical alignment are left, see Fig. 6(b). Then two alignments are conflicting if their corresponding edge segments cross or share a vertex. These conflicts are classified according to the number of involved inner edge segments. *Type* 2 *conflicts*, two crossing inner segments, are assumed to have been avoided by the crossing reduction phase and not to occur. For example this is automatically ensured by the barycenter and median methods. For sifting the absence of type 2 conflicts can be ensured by weighting each inner segment crossing with $|E|$ instead of 1. *Type* 1 *conflicts*, a non-inner segment crossing an inner segment, are resolved in favor of the inner segment. That means, the non-inner segment is removed from the graph. Finally, *type* 0 *conflicts*, two crossing non-inner segments, are resolved greedily in a leftmost fashion. That means, the right segment is removed from the graph. At this point there are no crossings left, see Fig. 6(c).

2) Horizontal Compaction: In the second step each maximum set of vertically aligned vertices, i. e., each connected component, is combined into a *block*, see Fig. 6(d). Consider the *block graph* obtained by introducing directed edges between each vertex and its successor (if any) on its level and by contracting the blocks into single vertices, see Fig. 6(e). A "horizontal" longest path layering on the block graph determines the x-coordinate of each block and thus of each contained vertex. Thereby the given minimum separation of the vertices δ is preserved.

The longest path layering leaves horizontal gaps between the blocks. Thus a further horizontal com-
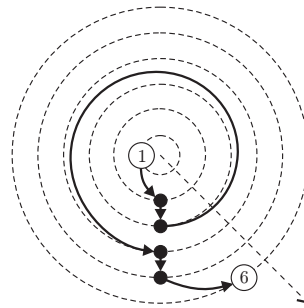

Fig. 7. Type 3 conflict

paction is possible: The block graph with expanded blocks is partitioned into *classes*, see Fig. 6(f). The first class is defined as the set of vertices which are reachable from the top left vertex. Then the class is removed from the block graph. This is repeated, until every vertex is in a class. Within the classes the graph is already compact. Now the algorithm places the classes as close as possible except for minimum separation δ. In Fig. 6(f) this already happened. Fig. 6(g) shows the complete left upper layout.

3) Balancing: At this point we have four x-coordinates for each vertex. The two left (right) aligned assignments are shifted horizontally so that their minimum (maximum) coordinate agrees with the minimum (maximum) coordinate of the smallest width layout. The resulting coordinate is the average median[4] of the four intermediate coordinates. After reinserting all removed segments, the resulting drawing is obtained, see Fig. 6(h).

B. Preprocessing

If an inner segment is a cut segment, i. e., if it crosses the ray, then the maximum of two bends for the corresponding long edge cannot be guaranteed, see Fig. 7 for an example. We call this situation a *type* 3 *conflict*. A simple solution is to demand the absence of inner cut segments in the input embedding, similar as it is done with type 2 conflicts. A different, more constructive and always doable approach described in the following, is to eliminate the conflicts by changing the position of the ray. This strategy changes the offset of some edges and thus changes the embedding. But this does not affect a later drawing.

Before we continue with the description of the elimination algorithm, we discuss an important property of radial level embeddings:

---

[4]If the median is not unique, the average median is defined as the average of the two median values.

(a) Level embedding

(b) Candidates

(c) Alignment

(d) Blocks

(e) Block graph

(f) Classes

(g) Left upper layout
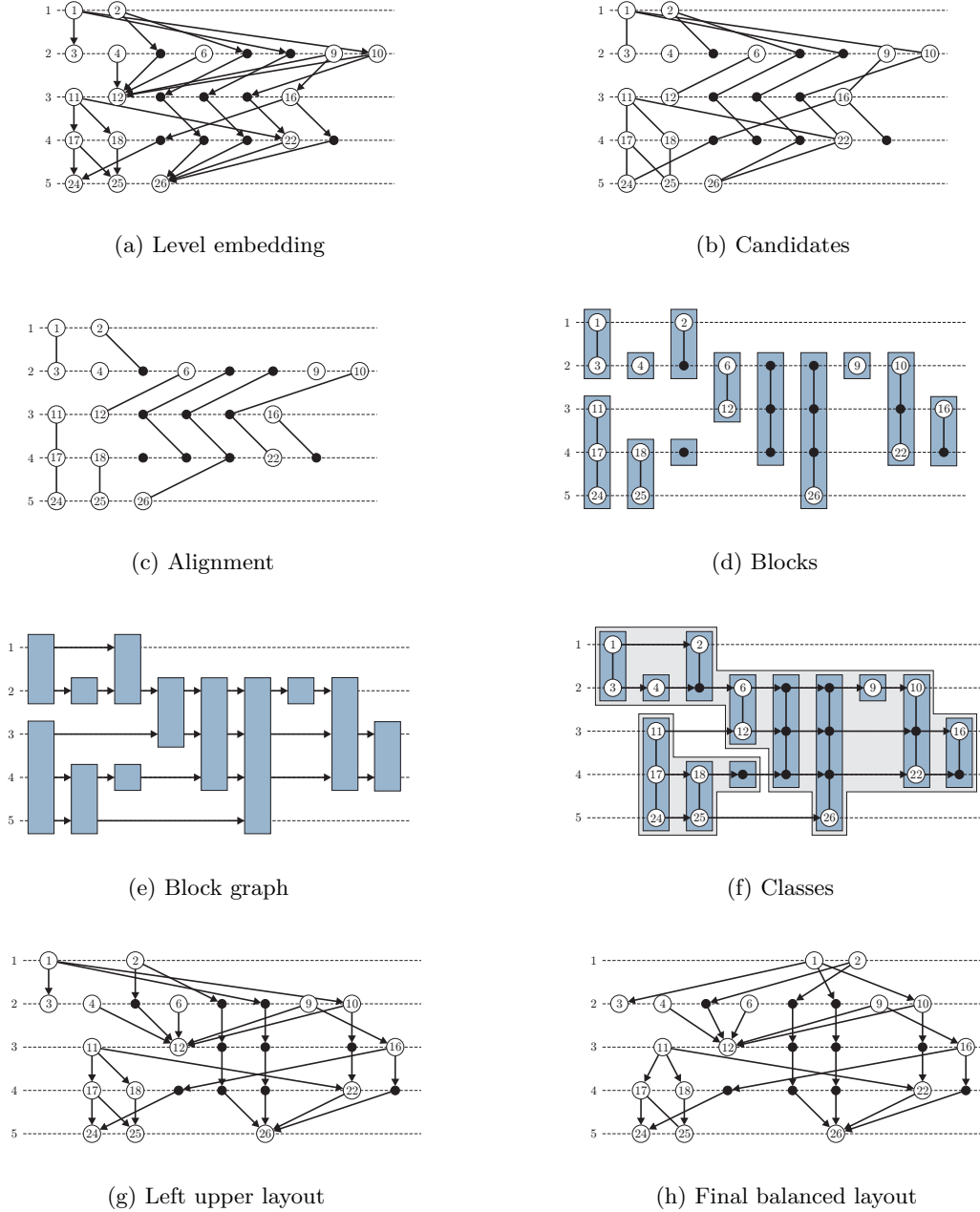

(h) Final balanced layout

Fig. 6.   Stages of the Brandes/Köpf algorithm

Lemma 4: Let $E_i' = \{ (u,v) \mid u \in B_{i-1}, v \in B_i \} \subseteq E$ be the set of all inner segments between levels $i-1$ and i with $2 < i < k$. Then for any two edges $e_1, e_2 \in E_i' : |\psi(e_1) - \psi(e_2)| \leq 1$.

Proof: In the extreme case let $e_1, e_2 \in E_i'$ for $2 < i < k$ be inner segments with $\psi(e_1) = \max\{\psi(e) \mid e \in E_i'\}$ and $\psi(e_2) = \min\{\psi(e) \mid e \in E_i'\}$. Now assume that $\psi(e_1) > \psi(e_2) + 1$. As a consequence e_1 and e_2 cross. This is a type 2 conflict and contradicts the absence of type 2 conflicts in the input embedding. ∎

In a first step to eliminate type 3 conflicts we consecutively *unwind* the levels in ascending order from 3 to $k-1$ with Algorithm 3. Between levels 1 and 2 resp. $k-1$ and k there are no inner segments. Clearly, level i is unwound by rotating the whole outer graph, i.e., all levels $\geq i$ are rotated by multiples of 360 degrees. Please note that UNWIND-LEVEL updates only offsets of edges between levels $i-1$ and i. The position of the ray, i.e., the ordering of the vertices, remains the same.

Lemma 5: After unwinding for each inner segment $e \in E : \psi(e) \in \{0, +1\}$.

Proof: Lemma 4 implies for each inner segment $e = (u,v)$ with $\phi(v) = i$ that $\psi(e) \leq 1$. Additionally

---

Algorithm 3: UNWIND-LEVEL

Input: Radial embedding $\mathcal{E} = (\pi, \psi)$ of a level graph
$G = (V \cup B, E, \phi)$ and level i with $2 < i < k$
Output: Updated embedding $\mathcal{E}$, i.e., offsets ψ of
inner segments entering level i

1 $m \leftarrow \min\{\psi(e) \mid e = (u, v) \in E, u \in B_{i-1}, v \in B_i\}$
2 foreach *segment* $e = (u, v) \in E$ *with* $v \in V_i \cup B_i$ **do**
3 $\quad\lfloor\ \psi(e) \leftarrow \psi(e) - m$

(a) Horizontal   (b) Radial

Fig. 8. Overlap of the left and right contour

$\psi(e)$ cannot be negative because we have subtracted the minimum over all inner segments entering level i. Since this argument holds for every level $2 < i < k$, the claim follows. ∎

Lemma 6: After unwinding there are no two dummy vertices $v, v' \in B_i$ on the same level i with $\psi\big((u, v)\big) = 0$, $\psi\big((u', v')\big) = +1$, and $v \prec v'$ for any $u, u' \in B_{i-1}$.

Proof: Follows directly from the absence of type 2 conflicts. ∎

Now we use rotation as described in Sect. IV to eliminate the remaining crossings of inner segments with the ray. Please note that rotation of a single level i is different from rotating levels during unwinding. Here we do not rotate by (multiples of) 360 degrees in general and do not rotate all levels $\geq i$ simultaneously. Let $B_i' \subseteq B_i$ be the set of dummy vertices incident to an incoming inner segment $e = (u, v)$ with $\psi(e) = +1$. Let $v = \arg\max\{\pi(v) \mid v \in B_i'\}$. We rotate level i clockwise until the ray enters the position after v, i.e., until v is the last vertex on i and thus $v = \arg\max\{\pi(v) \mid v \in V_i \cup B_i\}$. We use the clockwise direction, because according to Lemma 6 we do not generate new type 3 conflicts this way. Finally, all inner segments have an offset of 0. The overall running time is $\mathcal{O}(N)$.

C. Intermediate Horizontal Layout

In the next step we generate a horizontal layout of the radial level embedding with the Brandes/Köpf algorithm. Therefore we ignore all cut segments. Remember that the embedding is free of type 3 conflicts. Thus all inner segments of an edge are aligned vertically. The resulting layout will later be transformed into a concentric layout by concentrically connecting the ends of the horizontal level lines with their beginnings. Therefore, we must take into account that circumferences of radial level lines grow with ascending level numbers. Thus we use a minimum vertex separation distance $\delta_i = \frac{1}{i}$ for each horizontal level i, which is in each case indirectly proportional to i.
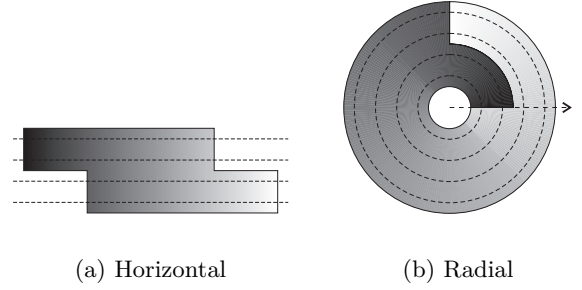
In this way we achieve a uniform minimum arc length between two neighbor vertices on every radial level line with the radial transformation described in the next section, since we use the level numbers $1, 2, \ldots, k$ as radii.

D. Radial Layout

At this stage every vertex $v \in V$ has Cartesian coordinates $\big(x(v), y(v) = \phi(v)\big) \in \mathbb{R} \times \mathbb{R}$. For the transformation into a radial drawing we interpret these coordinates as polar coordinates and transform them with Eq. (1) into Cartesian coordinates $\big(x_r(v), y_r(v)\big) \in \mathbb{R} \times \mathbb{R}$. The position of the ray denotes 0 degrees.

$$
\begin{aligned}
\big(x_r(v), y_r(v)\big) = \\
\Big(y(v) \cdot \cos\big(\tfrac{2\pi}{z} \cdot x(v)\big), y(v) \cdot \sin\big(\tfrac{2\pi}{z} \cdot x(v)\big)\Big)
\end{aligned} \tag{1}
$$

The factor $\frac{2\pi}{z}$ normalizes the length of the horizontal level lines to the circumferences of the radial level lines. We set $z = \max\big\{\max\{x(v') \mid v' \in V_i \cup B_i\} - \min\{x(v') \mid v' \in V_i \cup B_i\} + \delta_i \mid 1 \leq i \leq k\big\}$, i.e., z is the largest horizontal distance between two vertices on the same level i plus δ_i. The addend δ_i is necessary to maintain the minimum distance between the first and the last vertex, since they become neighbors on the radial level line. Let i_z be the level which defines z. The normalization automatically realizes the necessary overlap between the left and the right contour of the horizontal layout when drawn radially, see Fig. 8. Level i_z is the widest level and thus i_z defines the maximum overlap of the contours.

After drawing the vertices, we draw the edges as segments of a spiral. Each point p of a straight line segment $e = (u, v)$ is defined by Eq. (2) for $0 \leq t \leq 1$.

$$
\big(x(p), y(p)\big) = (1 - t)\big(x(u), y(u)\big) + t\big(x(v), y(v)\big) \tag{2}
$$

The coordinates of p can be transformed with Eq. (1). But e can be a cut segment, which winds multiple times clockwise or counter-clockwise around the center. Therefore we rather use Eq. (3) which simulates this behavior horizontally. Imagine $|\psi(e)|+1$ copies of the layout placed in a row, cf. Fig. 9. If $\psi(e) \geq 0$, then imagine e drawn as straight line from u in the leftmost layout to v in the rightmost layout. Otherwise, draw e from u in the rightmost layout to v in the leftmost one. Any two neighboring layouts of the row are separated by δ_{i_z}, i.e., the x-coordinate of the leftmost vertex in the right layout is the x-coordinate of the rightmost vertex of the left layout plus δ_{i_z}.

$$
\begin{aligned}
\big(x(p), y(p)\big) =& (1 - t)\big(x(u), y(u)\big) \\
& + t\big(x(v) + \psi(e) \cdot z, y(v)\big)
\end{aligned}
\tag{3}
$$

For all edges with offset 0 there is only one possible direction without crossing the ray, i.e., there is only one copy in the row. Equation (3) inserted in Eq. (1) for drawing a spiral segment between u and v results in the following equation:

$$
\begin{aligned}
\big(x_r(p), y_r(p)\big) = & \big((1 - t)y(u) + t \cdot y(v)\big) \\
& \cdot \bigg(\cos\Big(\tfrac{2\pi}{z} \cdot \big((1 - t)x(u) + t \cdot (x(v) + \psi(e) \cdot z)\big)\Big), \\
& \quad \sin\Big(\tfrac{2\pi}{z} \cdot \big((1 - t)x(u) + t \cdot (x(v) + \psi(e) \cdot z)\big)\Big)\bigg)
\end{aligned}
\tag{4}
$$

If $t = 0.5$, then p lies on a concentric circle with radius $\frac{\phi(u)+\phi(v)}{2}$, because the radius of the spiral segment grows proportional to the concentric distance between p and $\phi(u)$. To obtain smooth edges, the number of supporting points $s\colon E \to \mathbb{N}$ needed for drawing edges $e = (u, v)$ with an approximating polyline or spline depends on the edge length and a quality factor $Q \geq 1$.

$$
\begin{aligned}
s(e) & \sim \phi(v) \cdot \Big(\big|\tfrac{2\pi}{z} \cdot x(v) - \tfrac{2\pi}{z} \cdot x(u) + \psi(e) \cdot 2\pi\big|\Big) \cdot Q \\
& \sim \phi(v) \cdot \Big(\big|\tfrac{x(v)-x(u)}{z} + \psi(e)\big|\Big) \cdot Q
\end{aligned}
\tag{5}
$$

In the special case of $|V_1| = 1$, e.g., in Fig. 7, it is more aesthetic pleasing to place $v \in V_1$ into the concentric center, cf. Fig. 1(b). Thus we renumber the levels by $\phi'(w) = \phi(w) - 1$ for all $w \in V \cup B - \{v\}$, set $x_r(v) = y_r(v) = 0$, layout $G' = (V \cup B - \{v\}, E - \{(v, w) \mid w \in V\}, \phi')$, and draw each edge (v, w) as a straight line. To get a readable picture in the case $|V_1| > 1$, Eades [10] suggests to set the diameter of the first level to the radial distance between the radial

level lines. To achieve this with our algorithm, we use $0.5, 1.5, 2.5, \ldots, k-1.5, k-0.5$ as level numbers/radii. An equivalent solution is to double the number of levels $k' = 2k$, to renumber the levels by $\phi'(v) = 2\phi(v) - 1$ for all $v \in V \cup B$, to generate a drawing of $G' = (V \cup B, E, \phi')$, and finally to zoom by a factor of $\frac{1}{2}$.

Usually we draw on a canvas which has dimensions $a \times b$ and has the origin in the upper left corner. Thus for each vertex or supporting point p we do the following: With the translation $\big(x_r(p), y_r(p)\big) = \big(x_r(p)+\frac{a}{2}, y_r(p)+\frac{b}{2}\big)$ we move the origin to the center. In order to use the entire drawing space, we scale the layout by $\big(x(p), y(p)\big) = \big(x(p), y(p)\big) \cdot \frac{\min\{a,b\}}{2k}$.

Since the elimination of type 3 conflicts generates no new crossings and Eqs. (1) and 4 are bijective we do not change the crossing number given by the embedding. A radial level planar embedding is drawn planar. Adopting the common assumption that drawing a line (here an edge as a spiral segment with its supporting points) needs $\mathcal{O}(1)$ time, we obtain an $\mathcal{O}(N)$ running time.

VI. Conclusion

WE extended three well known crossing reduction techniques to radial level drawings. In practice, all algorithms are fast enough to be applied to reasonably large graphs. We showed by empirical evidence, that using radial instead of horizontal level lines reduces the number of crossings significantly. Further we have presented a new linear time algorithm for drawing level graphs (assigning coordinates) in a radial fashion. To check its performance and to visually confirm the good quality of the resulting drawings we realized a prototype as a plug-in for Gravisto [33] in Java. For a given embedding, the coordinates of a graph with $N = 50,000$ can be computed in less than 50 seconds on a 1.8 GHz PC with 768 MB RAM. For computing radial embeddings of graphs of this size sifting is too slow and one should choose the faster but qualitatively inferior barycenter or median method, analogously to the recommendation for horizontal embeddings.

Future research can address a more efficient sifting algorithm. Also, there are some interesting problems which we do not touch in this paper: Can the number of crossings $\chi(\mathcal{E})$ in a radial embedding $\mathcal{E}$ be computed in $\mathcal{O}(\chi(\mathcal{E}))$ time? Is $\chi(\mathcal{E}) \leq 3\chi(G)$ (or similar) for an embedding $\mathcal{E}$ computed by one-sided Cartesian median heuristic on a 2-level graph G as it is for horizontal median [17]? Are there efficient radial
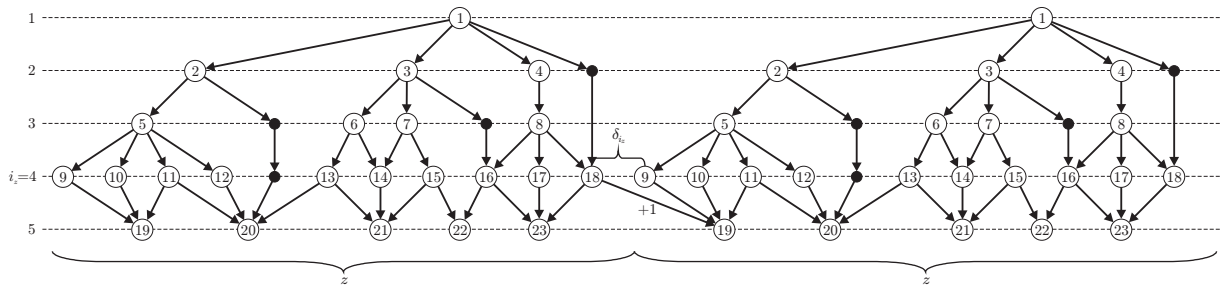
Fig. 9. Simulation of cut edges

extensions of other crossing reduction heuristics? A radial crossing reduction algorithm that already avoids type 3 conflicts in this phase would be helpful, since our elimination approach may create many crossings of non-inner segments with the ray.

ACKNOWLEDGMENT

The author would like to thank Michael Forster for contributing to the parent papers [15], [34] and Matthias Höllmüller for expunging some bugs and implementing our algorithms in Gravisto [33] within a network analysis package.

REFERENCES

[1] K. Sugiyama, S. Tagawa, and M. Toda, "Methods for visual understanding of hierarchical system structures," *IEEE Trans. Syst., Man, Cybern.*, vol. 11, no. 2, pp. 109–125, 1981.

[2] M. Kaufmann and D. Wagner, *Drawing Graphs*, ser. LNCS. Springer, 2001, vol. 2025.

[3] C. Bachmaier, F. J. Brandenburg, and M. Forster, "Radial level planarity testing and embedding in linear time," *Journal of Graph Algorithms and Applications*, vol. 9, no. 1, pp. 53–97, 2005.

[4] M. Baur and U. Brandes, "Crossing reduction in circular layout," in *Proc. Workshop on Graph-Theoretic Concepts in Computer Science, WG 2004*, ser. LNCS, J. Hromkovic, M. Nagl, and B. Westfechtel, Eds., vol. 3353. Springer, 2005, pp. 332–343.

[5] E. Mäkinen, "On circular layouts," *International Journal of Computer Mathematics*, vol. 24, pp. 29–37, 1988.

[6] J. M. Six and I. G. Tollis, "A framework and algorithms for circular drawings of graphs," *Journal of Discrete Algorithms*, vol. 4, no. 1, pp. 25–50, 2006.

[7] E. Di Giacomo, W. Didimo, L. Giuseppe, and H. Meijer, "Computing radial drawings on the minimum number of circles," *Journal of Graph Algorithms and Applications*, vol. 9, no. 3, pp. 347–364, 2005.

[8] M. Carpano, "Automatic display of hierarchized graphs for computeraided decision analysis," *IEEE Trans. Syst., Man, Cybern.*, vol. 10, no. 11, pp. 705–715, 1980.

[9] M. G. Reggiani and F. E. Marchetti, "A proposed method for representing hierarchies," *IEEE Trans. Syst., Man, Cybern.*, vol. 18, no. 1, pp. 2–8, 1988.

[10] P. Eades, "Drawing free trees," *Bulletin of the Institute of Combinatorics and its Applications*, vol. 5, pp. 10–36, 1992.

[11] U. Brandes, P. Kenis, and D. Wagner, "Centrality in policy network drawings," in *Proc. Graph Drawing, GD 1999*, ser. LNCS, J. Kratochvíl, Ed., vol. 1731. Springer, 1999, pp. 250–258.

[12] ——, "Communicating centrality in policy network drawings," *IEEE Trans. Visual. Comput. Graphics*, vol. 9, no. 2, pp. 241–253, 2003.

[13] T. J. Jankun-Kelly and K.-L. Ma, "Moiregraphs: Radial focus+context visualization and interaction for graphs with visual nodes," in *Proc. IEEE Symposium on Information Visualization, INFOVIS 2003*, T. Munzner and S. North, Eds. IEEE Computer Society Press, 2003, pp. 59–66.

[14] K.-P. Yee, D. Fisher, R. Dhamija, and M. Hearst, "Animated exploration of dynamic graphs with radial layout," in *Proc. IEEE Symposium on Information Visualization, INFOVIS 2001*, 2001, pp. 43–50.

[15] C. Bachmaier, F. Fischer, and M. Forster, "Radial coordinate assignment for level graphs," in *Proc. Computing and Combinatorics, COCOON 2005*, ser. LNCS, L. Wang, Ed., vol. 3595. Springer, 2005, pp. 401–410.

[16] E. G. Coffman and R. L. Graham, "Optimal scheduling for two processor systems," *Acta Informatica*, vol. 1, pp. 200–213, 1972.

[17] P. Eades and N. C. Wormald, "Edge crossings in drawings of bipartite graphs," *Algorithmica*, vol. 11, no. 1, pp. 379–403, 1994.

[18] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*. MIT Press, 2000.

[19] R. Rudell, "Dynamic variable ordering for ordered binary decision diagrams," in *Proc. IEEE/ACM International Conference on Computer Aided Design, ICCAD 1993*. IEEE Computer Society Press, 1993, pp. 42–47.

[20] C. Matuszewski, R. Schönfeld, and P. Molitor, "Using sifting for k-layer straightline crossing minimization," in *Proc. Graph Drawing, GD 1999*, ser. LNCS, J. Kratochvíl, Ed., vol. 1731. Springer, 1999, pp. 217–224.

[21] P. Eades and D. Kelly, "Heuristics for reducing crossings in 2-layered networks," *Ars Combinatorica*, vol. 21, no. A, pp. 89–98, 1986.

[22] M. Jünger and P. Mutzel, "2-layer straightline crossing minimization: Performance of exact and heuristic algorithms," *Journal of Graph Algorithms and Applications*, vol. 1, no. 1, pp. 1–25, 1997.

[23] U. Brandes and B. Köpf, "Fast and simple horizontal coordinate assignment," in *Proc. Graph Drawing, GD 2001*, ser. LNCS, P. Mutzel, M. Jünger, and S. Leipert, Eds., vol. 2265. Springer, 2001, pp. 31–44.

[24] C. Buchheim, M. Jünger, and S. Leipert, "A fast layout algorithm for k-level graphs," in *Proc. Graph Drawing, GD*

2000, ser. LNCS, J. Marks, Ed., vol. 1984. Springer, 2001, pp. 229–240.

[25] P. Eades, X. Lin, and R. Tamassia, "An algorithm for drawing hierarchical graphs," *International Journal of Computational Geometry & Applications*, vol. 6, pp. 145–156, 1996.

[26] P. Eades and K. Sugiyama, "How to draw a directed graph," *Journal of Information Processing*, vol. 13, no. 4, pp. 424–437, 1990.

[27] E. R. Gansner, E. Koutsofios, S. North, and K.-P. Vo, "A technique for drawing directed graphs," *IEEE Transactions on Software Engineering*, vol. 19, no. 3, pp. 214–230, 1993.

[28] E. R. Gansner, S. C. North, and K.-P. Vo, "DAG, a program that draws directed graphs," *Software – Practice and Experience*, vol. 17, no. 1, pp. 1047–1062, 1988.

[29] G. Sander, "Graph layout through the VCG tool," in *Proc. Graph Drawing, GD 1994*, ser. LNCS, R. Tamassia and I. G. Tollis, Eds., vol. 894. Springer, 1995, pp. 194–205.

[30] ——, "A fast heuristic for hierarchical Manhattan Layout," in *Proc. Graph Drawing, GD 1995*, ser. LNCS, F. J. Brandenburg, Ed., vol. 1027. Springer, 1996, pp. 447–458.

[31] ——, "Graph layout for applications in compiler construction," *Theoretical Computer Science*, vol. 217, pp. 175–214, 1999.

[32] P. Eades, Q.-W. Feng, and X. Lin, "Straight-line drawing algorithms for hierarchical graphs and clustered graphs," in *Proc. Graph Drawing, GD 1996*, ser. LNCS, S. C. North, Ed., vol. 1190. Springer, 1997, pp. 113–128.

[33] "Gravisto. Graph Visualization Toolkit," http://gravisto.fmi.uni-passau.de/, University of Passau.

[34] C. Bachmaier and M. Forster, "A radial adaption of the Sugiyama framework for hierarchical graph drawing," University of Passau, Tech. Rep. MIP-0603, April 2006.

Appendix
Crossing Reduction Benchmark Results

Figure 10 provides benchmarks comparing horizontal barycenter (HB), horizontal median (HM), and horizontal sifting (HS) crossing reduction heuristics with their radial variants, i.e., Cartesian barycenter (CB), Cartesian median (CM), and radial sifting (RS). The diagrams show that radial sifting is the best algorithm leaving dramatically fewer crossings than the others, but it is also the slowest, and that radial barycenter is the fastest. The same facts hold for the corresponding horizontal algorithms, which is folklore.

Christian Bachmaier received a diploma and a Doctorate in Computer Science from the University of Passau in 2000 and 2004 respectively. He now has a postdoctoral appointment at the University of Passau in the group Prof. Dr. Brandenburg. His research interests include algorithm engineering, computational complexity, data structures, efficient graph algorithms, graph drawing for the visualization of information, and graph planarity.
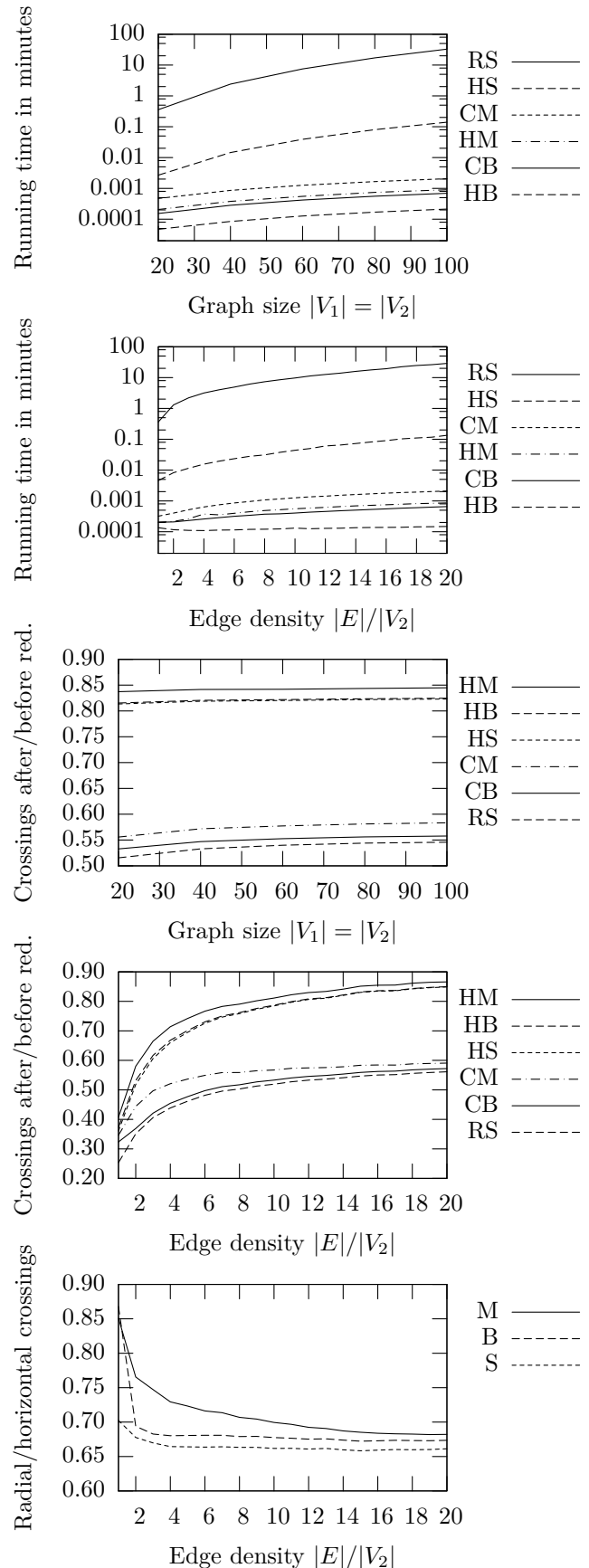

Fig. 10. Benchmarks