Radial Coordinate Assignment for Level Graphs*

Christian Bachmaier[1], Florian Fischer[2], and Michael Forster[3]

[1] University of Konstanz, 78457 Konstanz, Germany
christian.bachmaier@uni-konstanz.de
[2] Projective Software GmbH, 81543 Munich, Germany
florian.fischer@projective.de
[3] National ICT Australia, Eveleigh NSW 1430, Australia
michael.forster@nicta.com.au

Abstract. We present a simple linear time algorithm for drawing level graphs with a given ordering of the vertices within each level. The algorithm draws in a radial fashion without changing the vertex ordering, and therefore without introducing new edge crossings. Edges are drawn as sequences of spiral segments with at most two bends.

1 Introduction

In hierarchical graph layout, vertices are usually placed on parallel horizontal lines, and edges are drawn as polylines, which may bend when they intersect a level line. The standard drawing algorithm [9] 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).

We are especially interested in coordinate assignment. This phase is usually constrained not to change the vertex orderings computed previously. Further, it should support commonly accepted aesthetic criteria, 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. The novelty of this paper is to draw the level lines not as parallel horizontal lines, but as concentric circles, see Fig. 1. The apparent advantage is that level graphs can be drawn with fewer edge crossings. More level graphs can be drawn without any crossing at all, i. e., planar. Radial level drawings are common, e. g., in the study of social networks [2]. There vertices model *actors* and edges represent *relations* between them. The importance (*centrality*) of an actor is expressed by closeness to the concentric center, i. e., by a low level.

2 Preliminaries

A *k-level graph* $G = (V, E, \phi)$ with $k \leq |V|$ is a graph with a level assignment $\phi\colon V \to \{1, 2, \ldots, k\}$ that partitions the vertex set into k pairwise disjoint subsets

* Part of this work was done at the University of Passau.

L. Wang (Ed.): COCOON 2005, LNCS 3595, pp. 401–410, 2005.

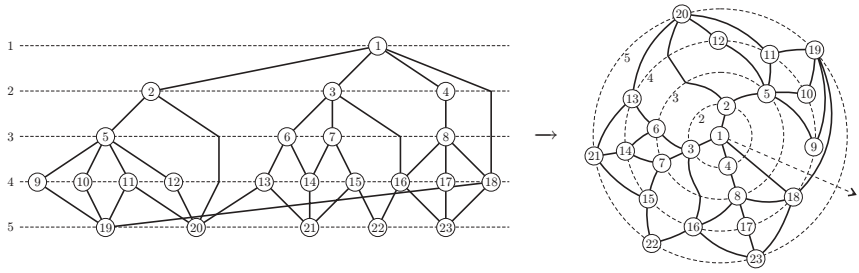


Fig. 1. Drawings of a level graph

$V = V_1 \dot\cup \cdots \dot\cup V_k$, $V_i = \phi^{-1}(i)$, $1 \le i \le k$, such that $\phi(u) \neq \phi(v)$ for each edge $(u, v) \in E$. Regardless whether G is directed or undirected, an edge incident to $u, v \in V$ is denoted by (u, v) if $\phi(u) < \phi(v)$. 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. 2(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 an *inner segment*. Let $N = |V \cup B| + |E|$ denote the size of the proper level graph $G = (V \cup B, E, \phi)$ where V contains the original vertices and $B = B_1 \dot\cup \ldots \dot\cup B_k$ contains the dummy vertices with $|B| \le k|E| \le |V||E|$. 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 we only consider proper level graphs $G = (V \cup B, E, \phi)$ in the following.

An *ordering* of a level graph is a partial order $\prec$ of $V \cup B$ such that $u \prec v$ or $v \prec u$ iff $\phi(u) = \phi(v)$. Define the (not necessarily consecutive) *positions* of the vertices as a function $\pi\colon V \cup B \to \mathbb{Z}$ with $u \prec v \Leftrightarrow \pi(u) < \pi(v)$ for any two vertices $u, v \in V_i \cup B_i$ on the same level i. In case that an ordering $\prec$ admits a level drawing of a level graph without edge crossings, $\prec$ is also called *level planar embedding*. We call an ordering of a level graph also level embedding.

3 Related Work

There are several algorithms for horizontal coordinate assignment using different approaches for the optimization of various objective functions or iterative improvement techniques, see [7] for an overview. Most interesting is the Brandes/Köpf algorithm [3], 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 Ø(N) running time and is fast in practice. For level planar embeddings Eades et al. [5] presented an algorithm that does not generate bends at all. However, it may need exponential area.

3.1 Horizontal Coordinate Assignment

Since the horizontal drawing algorithm of Brandes/Köpf [3] is the basis of our algorithm, we give an extended overview. Its first two steps are carried out four times and the third step combines the results.

Vertical Alignment. The objective is to align each vertex with its left upper, right upper, left lower, and right lower median neighbor. We only describe the alignment to the left upper median, the other three passes are analogous. First, all segments are removed that do not lead to an upper median neighbor, see Fig. 2(b). Then two alignments are conflicting if their edge segments cross or share a vertex. *Type* 2 *conflicts*, two crossing inner segments, are assumed to have been avoided by the crossing reduction phase and not to occur, e. g., using the barycenter method [7]. *Type* 1 *conflicts*, a non-inner segment crossing an inner segment, are resolved in favor of the inner segment, i. e., the non-inner segment is removed from the graph. *Type* 0 *conflicts*, two crossing non-inner segments, are resolved greedily in a leftmost fashion, i. e., the right segment is removed from the graph. At this point there are no crossings left, see Fig. 2(c).

Horizontal Compaction. Each maximum set of aligned vertices, i. e., each connected component, is combined into a *block*, see Fig. 2(d). Consider the *block graph* obtained by introducing directed edges between each vertex and its successor (if any) on its level, see Fig. 2(e). A "horizontal" longest path layering[1] determines the x-coordinate of each block and thus of each contained vertex. Thereby the given minimum vertex separation δ is preserved. The block graph with expanded blocks is partitioned into *classes*, see Fig. 2(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. In Fig. 2(f) this already happened. Fig. 2(g) shows the complete left upper layout.

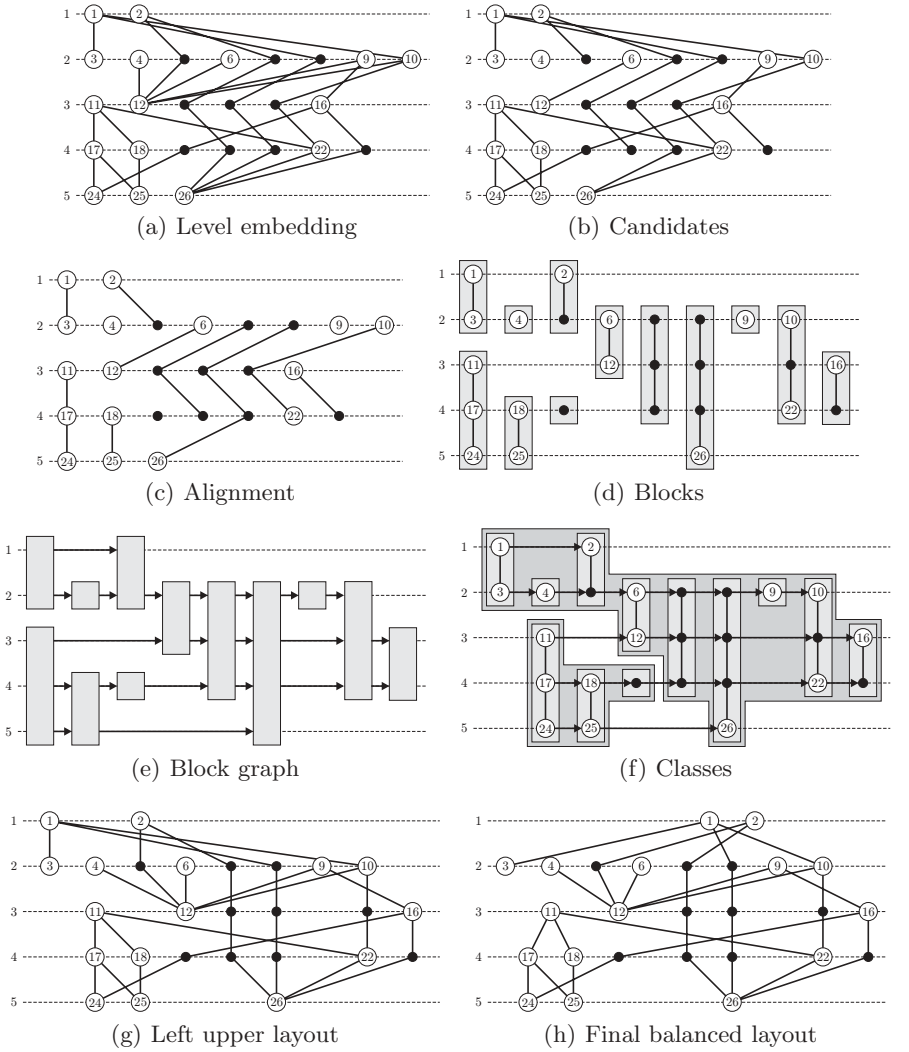
Balancing. Now each vertex has four x-coordinates. The two left (right) aligned assignments are shifted horizontally so that their minimum (maximum) coordinate agrees with the minimum (maximum) coordinate of the smallest width layout. The resulting coordinate is the average median[2] of the four intermediate coordinates. Fig. 2(h) shows the resulting drawing.

3.2 Radial Drawings

As we have already seen in Fig. 1, radial level lines help to reduce crossings. A further property of radial level lines is that they get longer with ascending level

[1] In a longest path layering vertices are assigned to levels: Each source of the graph is assigned to level 1. After removing all outgoing edges from this vertices, all (new) sources are assigned to level 2, and so on.

[2] The average median is defined as the average of the possible median values.



(a) Level embedding (b) Candidates

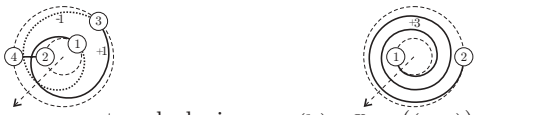
(c) Alignment (d) Blocks

(e) Block graph (f) Classes

(g) Left upper layout (h) Final balanced layout

Fig. 2. Stages of the Brandes/Köpf algorithm

number, which is especially useful if the sizes of the levels of the graph, i. e., the number of vertices, grow with the level number. The radial level idea originates from the ring diagrams of Reggiani and Marchetti [8] and from the radial tree drawings of Eades [4], which are common for free trees. There is an algorithm to compute a radial level planar embedding if one exists [1]. Thus, to maintain planarity, we require the preservation of the vertex ordering of the embedding in any case. However, our algorithm can draw arbitrary level embeddings.



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

(b) offset $\big((1, 2)\big) = +3$

Fig. 3. Offsets of edges

4 Radial Coordinate Assignment

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.

For radial level embeddings it is not only necessary to know the vertex ordering, but also where the ordering starts and ends on each level. Therefore we introduce a *ray* (the dashed arrow in Fig. 1) which tags this border between the vertices. The ray is a straight halfline from the concentric center to infinity. Since the embedding maintains the position $\pi(v)$ for every vertex $v \in V \cup B$, the position of the ray is implicitly evident. We call all edges intersecting the ray *cut segments*. For a radial level embedding it is not only important to know whether an edge is a cut segment, it is also necessary to know the direction of a cut segment, clockwise or counterclockwise from inner to outer level. Otherwise the drawing is not unique, see Fig. 3(a). Further, for uniqueness it is also important to know how many times a cut segment is wound around the concentric center, i. e., how many times it crosses the ray. Since the crossing reduction phase resp. the level planar embedding algorithm mentioned in Sect. 3.2 is aware of all this information, we assume it to be given as a function $\text{offset}\colon E \to \mathbb{Z}$ with the embedding. Thereby $|\,\text{offset}(e)|$ counts the crossings of e with the ray. If $\text{offset}(e) < 0$ ($\text{offset}(e) > 0$), e is a *clockwise* (*counter clockwise*) cut segment, i. e., the sign of $\text{offset}(e)$ reflects the mathematical direction of rotation, see Fig. 3(b). If $\text{offset}(e) = 0$, e is not a cut segment and thus needs no direction information. Observe that a cut segment cannot cross the ray clockwise and counter clockwise simultaneously.

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 [3] 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. 3.1 is not needed.

4.1 Preprocessing

If an inner segment is a cut segment, a radial alignment of the corresponding dummy vertices cannot be guaranteed, since each inner cut segment raises the

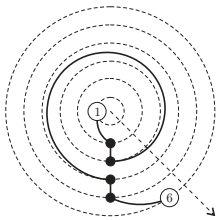


Fig. 4. Type 3 conflict

number of bends by two, see Fig. 4. 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 to the 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:

Lemma 1. *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* $e_1, e_2 \in E'_i : |\operatorname{offset}(e_1) - \operatorname{offset}(e_2)| \leq 1$.

Proof. In the extreme case let $e_1, e_2 \in E'_i$ for $2 < i < k$ be inner segments with $\operatorname{offset}(e_1) = \max\{\operatorname{offset}(e) \mid e \in E'_i\}$ and $\operatorname{offset}(e_2) = \min\{\operatorname{offset}(e) \mid e \in E'_i\}$. Now assume $\operatorname{offset}(e_1) > \operatorname{offset}(e_2)+1$. 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. □

In a first step to eliminate type 3 conflicts we consecutively *unwind* the levels in ascending order from 3 to $k-1$ with Algorithm 1. 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°. 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, stays the same.

Algorithm 1: UNWIND-LEVEL

Input: Embedding of $G = (V \cup B, E, \phi)$ and level i with $2 < i < k$
Output: Updated offsets of inner segments entering level i

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

Lemma 2. *After unwinding for each inner segment* $e \in E$: $\operatorname{offset}(e) \in \{0, +1\}$.

Proof. Lemma 1 implies for each inner segment $e = (u,v)$ with $\phi(v) = i$ that $\operatorname{offset}(e) \leq 1$. Additionally $\operatorname{offset}(e)$ cannot be negative since 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 3. *After unwinding there are no two dummy vertices $v, v' \in B_i$ on the same level i with* offset$\big((u,v)\big) = 0$, offset$\big((u',v')\big) = +1$, *and $v \prec v'$ for any $u, u' \in B_{i-1}$.*

Proof. This follows directly from the absence of type 2 conflicts. □

Contrary to horizontal layouts we have another freedom in radial layouts without changing the crossing number: *rotation* of a single level i. A clockwise rotation, cf. Algorithm 2, is moving the vertex v with the minimum position on the ordered level $\phi(v) = i$ over the ray by setting $\pi(v)$ to a new maximum on i. A counter clockwise rotation is symmetric. 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 segments incident to v must be changed. If rotating clockwise, the offsets of incoming segments of v are reduced by 1 and the offsets of outgoing segments are increased by 1. The offset updates for rotating counter clockwise are symmetric. Please note that rotation of a single level i is different to rotating levels within unwinding mentioned earlier. Here we do not rotate by (multiples of) 360° in general and do not rotate all levels $\geq i$ simultaneously.

Algorithm 2: ROTATE-CLOCKWISE

Input: Embedding of $G = (V \cup B, E, \phi)$ and level i with $1 \leq i \leq k$
Output: Updated offsets of segments entering or leaving level i

let $v = \text{argmin}\{\pi(v) \mid v \in V_i \cup B_i\}$
$\pi(v) \leftarrow \max\{\pi(v') \mid v' \in V_i \cup B_i\} + 1$
foreach *segment* $e = (u, v) \in E$ **do** offset$(e) \leftarrow$ offset$(e) - 1$
foreach *segment* $e = (v, w) \in E$ **do** offset$(e) \leftarrow$ offset$(e) + 1$

Rotation allows us to eliminate the remaining crossings of inner segments with the ray: Let $B'_i \subseteq B_i$ be the set of dummy vertices incident to an incoming inner segment $e = (u, v)$ with offset$(e) = +1$. Let $v = \text{argmax}\{\pi(v) \mid v \in B'_i\}$. 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 $v = \text{argmax}\{\pi(v) \mid v \in V_i \cup B_i\}$. We use the clockwise direction, because according to Lemma 3 we do not generate new type 3 conflicts this way. Finally, all inner segments have an offset of 0. The overall running time is $\mathcal{O}(N)$.

4.2 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. Since the embedding is type 3 conflict free, all inner segments 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 $\delta_i \leftarrow \frac{1}{i}$ for each horizontal level i, which is in each case indirect 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.

4.3 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 (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°.

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

The factor $\frac{2\pi}{z}$ normalizes the length of the horizontal level lines to the circumferences of the radial level lines. We set $z \leftarrow \max\big\{\max\{x(v') \mid v' \in V_i \cup B_i\} - \min\{x(v') \mid v' \in V_i \cup B_i\} + \delta_i \mid 1 \leq i \leq k\big\}$, 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 layout when drawn radially, see Fig. 5. Level i_z is the widest level and thus i_z defines the maximum overlap.

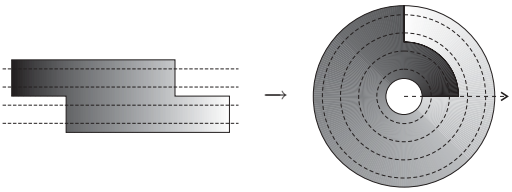


Fig. 5. Overlap of the left and right contour

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 (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 (1). But e can be a cut segment, which winds multiple times clockwise or counter clockwise around the center. Therefore we rather use (3) which simulates this behavior horizontally. Imagine

$|\operatorname{offset}(e)|+1$ copies of the layout placed in a row. If $\operatorname{offset}(e) \geq 0$, 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}.

$$\big(x(p), y(p)\big) = (1-t)\big(x(u), y(u)\big) + t\big(x(v) + \operatorname{offset}(e) \cdot z, y(v)\big) \tag{3}$$

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 (1) for drawing a spiral segment between u and v results in the following equation:

$$\begin{aligned}\big(x_r(p), y_r(p)\big) \leftarrow \big((1-t)y(u) + t \cdot y(v)\big) \cdot \\ \left(\cos\left(\tfrac{2\pi}{z} \cdot \big((1-t)x(u) + t \cdot (x(v) + \operatorname{offset}(e) \cdot z)\big)\right),\right. \\ \left.\sin\left(\tfrac{2\pi}{z} \cdot \big((1-t)x(u) + t \cdot (x(v) + \operatorname{offset}(e) \cdot z)\big)\right)\right)\end{aligned} \tag{4}$$

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 get smooth edges, the number of needed supporting points $s\colon E \to \mathbb{N}$ for drawing edges $e = (u, v)$ with an approximating polyline or spline depends on the edge length and a quality factor $q \geq 1$.

$$\begin{aligned} s(e) &\sim \phi(v) \cdot \left(\left|\tfrac{2\pi}{z} \cdot x(v) - \tfrac{2\pi}{z} \cdot x(u) + \operatorname{offset}(e) \cdot 2\pi\right|\right) \cdot q \\ &\sim \phi(v) \cdot \left(\left|\tfrac{x(v)-x(u)}{z} + \operatorname{offset}(e)\right|\right) \cdot q \end{aligned} \tag{5}$$

In the special case of $|V_1| = 1$ it is more aesthetical to place $v \in V_1$ into the concentric center, cf. Fig. 1. Thus we renumber the levels by $\phi'(w) \leftarrow \phi(w) - 1$ for all $w \in V \cup B - \{v\}$, set $x_r(v) \leftarrow y_r(v) \leftarrow 0$, layout $G' = (V \cup B - \{v\}, E - \{\,(v, w) \mid w \in V\,\}, \phi')$, and draw each edge (v, w) as straight line. In order to get a harmonic picture in the case $|V_1| > 1$, Eades [4] 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.

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 $\big(x_r(p), y_r(p)\big) \leftarrow \big(x_r(p) + \frac{a}{2}, y_r(p) + \frac{b}{2}\big)$ we move the origin to the center. In order to use the entire drawing space, we scale the layout by $\big(x(p), y(p)\big) \leftarrow \big(x(p), y(p)\big) \cdot \frac{\min\{a,b\}}{2k}$.

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

5 Conclusion

We have presented a new linear time algorithm for drawing level graphs 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 the Gravisto project [6] in Java. 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 using Java2 1.4.

Further investigations are required for radial crossing reduction algorithms that avoid type 3 conflicts already at this stage, since our elimination approach may create many crossings of non inner segments with the ray. The crossing reduction should also minimize the absolute values of the edge offsets, since this reduces crossings in general. It may be reasonable to reduce the angles spanned by the edges at the expense of a slightly increasing crossing number.

References

1. C. Bachmaier, F. J. Brandenburg, and M. Forster. Radial level planarity testing and embedding in linear time. *Journal of Graph Algorithms and Applications*, 2005. To appear. Preprint under `http://www.infosun.fmi.uni-passau.de/~chris/`.
2. U. Brandes, P. Kenis, and D. Wagner. Communicating centrality in policy network drawings. *IEEE Transact. Vis. Comput. Graph.*, 9(2):241–253, 2003.
3. U. Brandes and B. Köpf. Fast and simple horizontal coordinate assignment. In P. Mutzel, M. Jünger, and S. Leipert, editors, *Proc. Graph Drawing, GD 2001*, volume 2265 of *LNCS*, pages 31–44. Springer, 2001.
4. P. Eades. Drawing free trees. *Bulletin of the Institute of Combinatorics and its Applications*, 5:10–36, 1992.
5. P. Eades, Q.-W. Feng, and X. Lin. Straight-line drawing algorithms for hierarchical graphs and clustered graphs. In S. C. North, editor, *Proc. Graph Drawing, GD 1996*, volume 1190 of *LNCS*, pages 113–128. Springer, 1997.
6. Gravisto. Graph Visualization Toolkit. `http://www.gravisto.org/`. University of Passau.
7. M. Kaufmann and D. Wagner. *Drawing Graphs*, volume 2025 of *LNCS*. Springer, 2001.
8. M. G. Reggiani and F. E. Marchetti. A proposed method for representing hierarchies. *IEEE Transact. Systems, Man, and Cyb.*, 18(1):2–8, 1988.
9. K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierarchical system structures. *IEEE Trans. Systems, Man, and Cyb.*, 11(2):109–125, 1981.