Discrete Applied Mathematics 158 (2010) 159–179

Contents lists available at ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

Crossing minimization in extended level drawings of graphs

Christian Bachmaier [a,*], Hedi Buchner [b], Michael Forster [a], Seok-Hee Hong [c]

[a] University of Passau, Faculty of Inf. and Math., 94030 Passau, Germany
[b] IMAGEN Program, National ICT Australia, Eveleigh, NSW 1430, Australia
[c] School of Information Technologies, University of Sydney, NSW 2006, Australia

ARTICLE INFO

Article history:
Received 13 August 2008
Received in revised form 12 March 2009
Accepted 2 September 2009
Available online 19 September 2009

Keywords:
(Radial) level graph
Crossing minimization
Intra-level edges
Hierarchy
Level/layered drawing
Visualization of social networks
Graph algorithm

ABSTRACT

The most popular method of drawing directed graphs is to place vertices on a set of horizontal or concentric levels, known as level drawings. Level drawings are well studied in Graph Drawing due to their strong application for the visualization of hierarchy in graphs. There are two drawing conventions: Horizontal drawings use a set of parallel lines and radial drawings use a set of concentric circles.

In level drawings, edges are only allowed between vertices on different levels. However, many real world graphs exhibit hierarchies with edges between vertices on the same level. In this paper, we initiate the new problem of extended level drawings of graphs, which was addressed as one of the open problems in social network visualization, in particular, displaying centrality values of actors. More specifically, we study minimizing the number of edge crossings in extended level drawings of graphs. The main problem can be formulated as the extended one-sided crossing minimization problem between two adjacent levels, as it is folklore with the one-sided crossing minimization problem in horizontal drawings.

We first show that the extended one-sided crossing minimization problem is $\mathcal{NP}$-hard for both horizontal and radial drawings, and then present efficient heuristics for minimizing edge crossings in extended level drawings. Our extensive experimental results show that our new methods reduce up to 30% of edge crossings.

1. Introduction

A *level drawing* (or *hierarchical drawing*) of a graph is the most popular drawing convention for directed graphs, alternatively known as the *Sugiyama method* [24]. Consequently, drawing level graphs is a well-studied problem in Graph Drawing. There is a rich literature on drawing level graphs including characterizations of level-planar graphs, level planarity testing, crossing minimization, and planarization methods for non-level planar graphs, see [17].

There are two drawing conventions for level graphs: in *horizontal drawings*, vertices are placed on parallel horizontal lines and edges are drawn as strictly y-monotone polylines that may bend when they intersect a level line [24,17,11]. In *radial drawings*, vertices are placed on concentric circles and edges are drawn as polyline segments of spirals which are monotone from the concentric center to the outside [1]. Both drawings are produced based on the same drawing framework, the Sugiyama method, which consists of the following four steps:

(1) *Cycle removal*: Reverse appropriate edges to eliminate cycles.

* Corresponding author. Tel.: +49 851 509 3035; fax: +49 851 509 3032
E-mail addresses: bachmaier@fim.uni-passau.de (C. Bachmaier), hedi.buchner@nicta.com.au (H. Buchner), forster@fim.uni-passau.de (M. Forster), shhong@it.usyd.edu.au (S.-H. Hong).

0166-218X/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.dam.2009.09.002

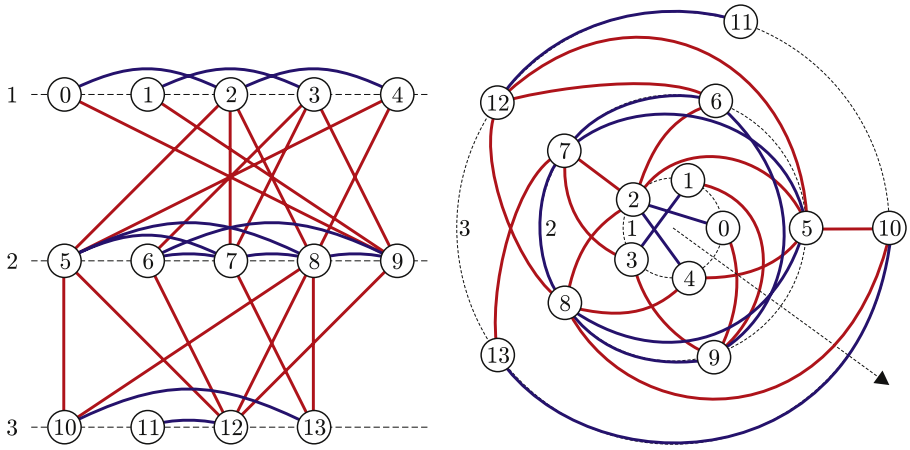


Fig. 1. Example drawings.

(2) *Level assignment*: Assign vertices to levels such that no edges have both end vertices on the same level, and introduce dummy vertices to represent long edges which span more than one level by a path of proper edges. The dummy vertices represent edge bends.
(3) *Crossing minimization*: Compute a good ordering of the vertices on each level to minimize edge crossings between two adjacent levels.
(4) *Coordinate assignment*: Assign x-/angular coordinates to the vertices to meet some esthetic criteria. The y-/radial coordinates are implicit through the levels.

However, many real world graphs exhibit hierarchies with edges between the vertices on the same level. For example, the visualization of *centrality* of *actors* in social networks produces level graphs with both *inter-level* and *intra-level* edges [9]. Note that up to now it is neither shown if intra-level edges reduce the visual complexity nor if they reduce the overall number of crossings. The fact that all existing hierarchical drawing methods more or less simply ignore intra-level edges, although they are present from the respective application in most cases, justifies an investigation. To our best knowledge, the only exception is the *compound graph* drawing algorithm of Sugiyama and Misue [23] where a fast but qualitatively inferior barycenter strategy on intra-level edges is used to avoid crossings of edges and bounding rectangles of a *compound node*.

We initiate the new problem of drawing *extended level graphs*, i.e. , level graphs with intra-level edges. Drawing extended level graphs was addressed as one of the open problems in social network visualization by Brandes [5]. The proposed goal is an easy human perception, where one of the main criteria seems to be an overall low number of crossings [21]. Extended level graphs often occur in practice, for example graphs where the level assignment is already predefined by a breadth first search to express distances, or social networks where the importance (centrality) of an actor (modeled by a vertex) defines its level [7,8,6]. More specifically, the visualization of an actor's status in a social network is a horizontal drawing where the levels are not equidistant, because each level represents a real-valued centrality index for the actors [9]. Since the centrality values often differ only marginally, status values can be clustered. The actors with centrality values in the same range are assigned to the same level to avoid perceptual problems of having too many levels. However, this approach introduces many intra-level edges. In the conclusion of [9], Brandes et al. state that the treatment of intra-level edges needs further investigation. Later, Brandes proposed a new research direction on minimizing all types of crossings including inter-level edge and intra-level edges, for social network visualization as an open problem [5].

Another application of extended level graphs with radial drawings is the visualization of *micro/macro graphs* [3], e.g. , arising from group analysis or role assignment in social networks [6]. In general, intra-level edges may help to gain better aspect ratios, since drawings tend to be much longer than wide, especially with the Sugiyama method.

In our *extended level drawings* of extended level graphs, we represent intra-level edges using semicircles with different radii in order to avoid overlapping edges and crossings between vertices and edges, see Fig. 1. Further, we restrict to drawing the semicircles only on one side of the level lines, say above (inside), in order to model the problem as in the Sugiyama framework.

In this paper, we study the crossing minimization problem in extended level graphs to improve the readability [21] of the extended level drawings. More precisely, we study the new problem of minimizing the number of edge crossings in extended level drawings of graphs. The main focus of this paper can be formally defined as the *extended one-sided crossing minimization* problem between two adjacent levels, similar to the well-known *one-sided crossing minimization* for the horizontal drawing convention. We show that the one-sided crossing minimization problem for extended level graphs is $\mathcal{NP}$-hard for both horizontal and radial drawings, and present greedy heuristics for minimizing edge crossings that take different types of

Table 1
Survey of treated edge crossing variants.

	Inter-level	Mixed	Intra-level	All
Horizontal drawings	[17]	✓	[4]/✓	✓
Radial drawings	[1]	✓	✓	✓

edge crossings into account: *inter-level crossings* between two inter-level edges, *intra-level crossings* between two intra-level edges, and *mixed crossings* between intra-level edges and inter-level edges. Note that greedy heuristics are state of the art in this area [17] since they are much faster and, thus, can treat larger graphs than common local search algorithms.

Our main aim is to extend the well-known *sifting* heuristic for level drawings. More specifically, we designed new *extended sifting* heuristics by carefully integrating sifting, radial sifting, and circular sifting methods together with a new crossing counting algorithm. Our extensive experimental results show that our new methods reduce up to 30% of crossings compared to existing standard heuristics which only consider inter-level edge crossings. Of course, this value is only a rule of thumb for reasonable ratios between inter-level and intra-level edges. Since our algorithm is the first one which does not simply ignore intra-level edge crossings, it is clear that the more intra-level edges are present, the higher the gain will be. The running times of our algorithms are within the same bounds as the traditional sifting algorithms. Thus, we are able to handle the same graph sizes within similar times.

The checkmarks in Table 1 summarize the problems solved in this paper, which is organized as follows: After explaining some necessary preliminaries in Section 2, we present our extended sifting heuristics for extended level graphs, which explicitly consider three different types of crossings in horizontal drawings in Section 3. Then, we present extended radial sifting heuristics for extended level graphs in radial drawings in Section 4. Section 5 presents our experimental results and Section 6 concludes with some open problems.

2. Preliminaries

A (*proper*) *k-level graph* $G = (V, E, \phi)$ is a graph with a level assignment $\phi: V \to \{1, 2, \ldots, k\}$, 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 $|\phi(u) - \phi(v)| = 1$ for each *inter-level edge* $\{u, v\} \in E$. Particularly, $k = 1$ implies that $E = \emptyset$. For $v \in V$ with $\phi(v) > 1$ let $E(v) = \{\{u, v\} \in E \mid u \in V_{\phi(v)-1}\}$ be the (predecessor) *inter-level adjacency list*. Define $E(v) = \emptyset$, if $\phi(v) = 1$. An *ordering* of a level graph is a partial order $\prec$ of V such that $u \prec v$ or $v \prec u$ iff $\phi(u) = \phi(v)$ for each pair of vertices $u, v \in V$. If the vertex sets V_i are ordered sets (according to $\prec$), we call G an *ordered level graph*.

2.1. Sifting with a crossing matrix

The most common technique for crossing minimization in level drawings is to only consider two consecutive levels at a time in multiple top-down and bottom-up passes. Starting with an arbitrary ordering of the first level, subsequently the ordering of one level is fixed, while the subsequent level is reordered to minimize the number of crossings in-between. Thus, the 2-level horizontal drawing is the fundamental building block for drawing level graphs with k-levels.

The well-studied *one-sided 2-level crossing minimization problem* is formally defined as follows: Given a 2-level graph $G = (V_1 \dot\cup V_2, E, \phi)$, where the vertex set V_1 is given with a fixed ordering, compute an ordering of V_2 which produces the minimum number of crossings. This is known to be $\mathcal{NP}$-hard [13] and a number of heuristics, approximation algorithms, and exact algorithms have been proposed. Eades and Wormald [13] proposed a *median heuristic*, which produces a 3-approximate solution to the one-sided crossing minimization problem. The *barycenter heuristic* by Sugiyama et al. [24] is an $O(\sqrt{n})$-approximation [13]. The barycenter (median) heuristic assigns each vertex of V_2 the barycenter (median) value of its neighbors in V_1, assuming the positions of vertices in V_1 are numbered from 1 to $|V_1|$ according to $\prec$. A sorting according to these values defines the ordering among the vertices in V_2. Currently, the best known approximation algorithm for the one-sided crossing minimization problem given by Nagamochi [20] delivers 1.4664-approximate solutions. Jünger et al. [15,16] presented integer linear programming algorithms and experimentally compared the exact results with various heuristics. See [11,17] for an extended overview.

For our new problem of one-sided crossing minimization in extended level graphs, we will adopt the sifting heuristic, which is slower than simple heuristics like barycenter or median heuristics, however, it produces fewer crossings in practice. Sifting was originally introduced as a heuristic for vertex minimization in ordered binary decision diagrams [22], and later adapted for the one-sided crossing minimization problem [19]. The main idea is to keep track of the objective function while moving in a *sifting step* a vertex $u \in V_2$ along with a fixed ordering of all other vertices in V_2 and then placing u to its locally optimal position. This is done by iteratively swapping consecutive vertices only.

The method is thus an extension of the *greedy-switch* heuristic [12], where u is swapped iteratively with its successor. We call a single swap a *sifting swap*. Executing a sifting step for every vertex in V_2 is called a *sifting round*. For crossing minimization, the objective function is the number of crossings between the edges incident to the vertex under consideration and all other edges. The efficient computation of the crossing count in sifting is based on the *crossing matrix*. The $|V_2|^2$ entries

in the crossing matrix correspond to the number of crossings caused by (the edges of) pairs of vertices in a particular relative ordering and can be computed as a preprocessing step in $\mathcal{O}(|E|^2)$ time [25,26]. Whenever a vertex is placed in a new position, only a small number of updates is necessary. This allows a running time of $\mathcal{O}(|V_2|^2)$ for one round. In practice, only few sifting rounds (3–5 for reasonable problem instances) are necessary to reach a local optimum for all vertices simultaneously. Our experiments showed that this is in most cases also the global optimum which we computed for small graphs with the ILP formulation of [16]. The largest reduction of crossings usually occurs in the first round.

2.2. Crossing minimization in radial drawings

Compared to the horizontal drawings of level graphs, radial drawings of level graphs have not been well studied. The problem of crossing minimization in radial drawing is more challenging, as it involves both vertex ordering and edge routing problems. That is, even if the orderings of vertices in both orbits are fixed, we still need to decide how to route (i.e. clockwise or counterclockwise) each edge around the inner orbit in order to minimize the number of edge crossings in a radial drawing.

Bachmaier [1] presented a new radial drawing framework, an adaptation of the Sugiyama method [24] to radial drawings. He proved that the one-sided crossing minimization problem in radial drawings is $\mathcal{NP}$-hard and presented a number of heuristics including *radial sifting* with experimental results. The first polynomial time 15-approximation algorithm for one-sided crossing minimization problem in radial drawings was presented by Hong and Nagamochi [14]. Their main contribution was to reduce a given instance of the one-sided crossing minimization in a radial drawing to that of the one-sided crossing minimization in a horizontal drawing.

As our new *extended radial sifting* heuristics for extended level graphs is based on radial sifting, we will explain details including basic terminologies in Section 4.

2.3. Circular sifting

The asymptotic overall running time of the original algorithm described above is $\mathcal{O}(|E|^2 + |V_2|^2)$ and too high for our purposes, i.e., to handle large graphs. Thus, we apply the *circular sifting heuristic* of Baur and Brandes [4] used for the $\mathcal{NP}$-hard [18] crossing minimization problem in circular drawings: Order the vertices V of a graph $G = (V, E)$ which all are placed on a single circle, e.g., as in Fig. 5, to minimize the number of crossings among the straight-line edges in E. Since there is no "circular" order, Baur and Brandes define linear orders $\prec_\alpha$ by selecting a reference vertex $\alpha \in V$ which is the first of the (here counterclockwise) sequence. For finding the locally optimal position of a vertex $u \in V$ in a sifting step, it is sufficient to record the change in crossing count while swapping u with its successor $v_p \in V$. This can be done by considering only edges incident to u or v_p: After a swap exactly those pairs of these edges cross which did not cross before. All other crossings remain unchanged (let $\chi(\pi)$ be the number of crossings of a drawing π and $N(v)$ be the set of adjacent vertices of $v \in V$).

Lemma 2.1 (*Baur, Brandes*). *Let $u \prec_u v_p \in V$ be consecutive vertices in a circular drawing π and let π' be the drawing with their positions swapped, then*

$$\chi(\pi') = \chi(\pi) - \sum_{x \in N(u)} |\{y \in N(v_p) \mid y \prec_x^{\pi} u\}| + \sum_{y \in N(v_p)} |\{x \in N(u) \mid x \prec_y^{\pi'} v_p\}|.$$

At the end of one step, u is placed where the intermediary crossing counts reached their minimum. For efficiency reasons, the computation of the change in crossing count is implemented over suffix lengths in ordered adjacency lists.

2.4. Inter-level sifting for crossings between inter-level edges

For horizontal level lines, we adapt the above idea to one-sided 2-level crossing minimization, which we call *inter-level sifting* for simplicity. We mainly exchange $\prec_\alpha$ by $\prec$ and virtually connect the start and the end points of the level lines to obtain a circle. Then, we only consider the ordering of the permutable level 2 as presented by Algorithms 1, 2 and 3. We obtain the same results as with the matrix method, without knowing the absolute crossing numbers, however. Since all three methods are generic and are also used for the following algorithms, Algorithm 2 already contains lines 4 and 8. At present, these lines can be ignored and the input graphs can be considered as $G = (V_1 \dot\cup V_2, E, \phi)$ for ease of understanding. For efficiency reasons, all shown operations are implemented in place on the graph data structure.

Algorithm 1. SIFTING-ROUND

Input: Ordered 2-level graph $G = (V_1 \dot\cup V_2, E, H, \phi)$
Output: Updated ordering of V_2

```
1 foreach u ∈ V2 do
2     V2 ← SIFTING-STEP(G, u)
3 return V2
```

Algorithm 2. SIFTING-STEP

Input: Ordered 2-level graph $G=(V_1 \dot\cup V_2, E, H, \phi)$, Vertex $u \in V_2$ to sift
Output: Updated ordering of V_2

```
1  let v_0 = u ≺ v_1 ≺ ··· ≺ v_{|V_2|-1} be the current ordering of V_2 with u put to front
2  foreach v ∈ V_2 do
3      Sort E(v) ⊆ E on ascending ordering of V_1 in O(|E|) time
4      Sort H_l(v), H_r(v) ⊆ H on ascending ordering of V_2 in O(|H|) time
5  χ ← 0; χ* ← 0                          // current and best number of crossings
6  p* ← 0                                 // best vertex position
7  for p ← 1 to |V_2| − 1 do
8      l ← UPDATE-INTRA-ADJ(G, u, v_p)
9      χ ← χ+SIFTING-SWAP-INTER(G, u, v_p)
10     if χ < χ* then
11         χ* ← χ; p* ← p
12 return V_2 ← v_1 ≺ ··· ≺ v_{p*-1} ≺ u ≺ v_{p*} ≺ ··· ≺ v_{|V_2|-1}
```

Algorithm 3. SIFTING-SWAP-INTER

Input: Ordered 2-level graph $G=(V_1 \dot\cup V_2, E, H, \phi)$, Swap vertices $u, v_p \in V_2$
Output: Change in crossing count

```
1  let x_0 ≺ ··· ≺ x_{r-1} be the neighbors of u in V_1
2  let y_0 ≺ ··· ≺ y_{s-1} be the neighbors of v_p in V_1
3  c ← 0; i ← 0; j ← 0
4  while i < r and j < s do
5      if x_i ≺ y_j then
6          c ← c + (s − j)
7          i ← i + 1
8      else if y_j ≺ x_i then
9          c ← c − (r − i)
10         j ← j + 1
11     else
12         c ← c + (s − j) − (r − i)
13         i ← i + 1; j ← j + 1
14 return c
```

3. Crossing minimization on horizontal levels

An *extended k*-level graph $G = (V, E, H, \phi)$ is a *k*-level graph (V, E, ϕ) which additionally has *intra-level edges* $\{u, v\} \in H$ with $\phi(u) = \phi(v)$. For $v \in V_i$ let $H_l(v) = \{\{u, v\} \in H \mid u \prec v\}$ be the *left intra-level adjacency list* and $H_r(v) = \{\{v, w\} \in H \mid v \prec w\}$ be the *right intra-level adjacency list*.

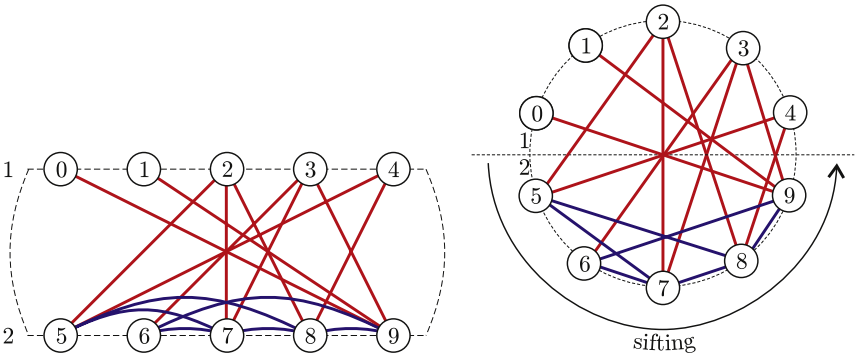
In this section, we first consider the new problem of one-sided 2-level crossing minimization for extended level graphs with horizontal drawings. It is easy to see that the one-sided 2-level crossing minimization problem for an extended 2-level graph is $\mathcal{NP}$-hard, since at least two subproblems, considering only inter-level edges [13] and considering only intra-level edges [18] are $\mathcal{NP}$-hard. The circular crossing minimization in [18] is exactly the same as minimizing crossings among intra-level edges of a horizontal level *i* (consider the level line *i* bent to a circle).

Lemma 3.1. *The one-sided 2-level crossing minimization problem for extended level graphs in horizontal drawings is $\mathcal{NP}$-hard.*

This motivates us to design efficient heuristics for the problem and we design extensions of the sifting heuristic for extended level graphs in order to compute a reasonable solution efficiently. After presenting a simple integrated method that only works for horizontal drawings, we then present our main method, extended sifting heuristics for horizontal drawings, which consists of three subroutines, each minimizing different types of crossings.

3.1. Compact method

The extended one-sided 2-level crossing minimization problem on horizontal levels can be transformed into a corresponding circular crossing minimization problem where the vertices of the two levels are placed on the two disjoint



(a) 2-level graph. (b) Problem transformation.

Fig. 2. Using circular sifting for extended one-sided 2-level crossing minimization.

semicircles, i.e., vertices on level 1 clockwise on semicircle 1 and vertices on level 2 counterclockwise on semicircle 2, see Fig. 2. Then, only vertices in V_2 are sifted using positions in semicircle 2 only. This is a minor modification of the original circular sifting.

Definition 3.1. For a 2-level graph $G = (V_1 \dot\cup V_2, E, H, \phi)$ let π be a horizontal drawing with

$$u_1 \prec^\pi u_2 \prec^\pi \cdots \prec^\pi u_{r=|V_1|} \quad \text{and} \quad v_1 \prec^\pi v_2 \prec^\pi \cdots \prec^\pi v_{|V_2|}$$

for vertices u_i, $r \in V_1$, $1 \le i \le |V_1|$, and $v_j \in V_2$, $1 \le j \le |V_2|$.

A *corresponding circular drawing* π' is a circular drawing of G with

$$u_{|V_1|} \prec_r^{\pi'} u_{|V_1|-1} \prec_r^{\pi'} \ldots \prec_r^{\pi'} u_1 \prec_r^{\pi'} v_1 \prec_r^{\pi'} v_2 \prec_r^{\pi'} \cdots \prec_r^{\pi'} v_{|V_2|-1}.$$

Lemma 3.2. *Let π' be a corresponding circular drawing of a 2-level graph $G = (V_1 \dot\cup V_2, E, H, \phi)$ drawing π. Then two edges cross in π' if and only if they cross in π.*

Proof. Let $e_1 = (u_1, v_1), e_2 = (u_2, v_2) \in E$ be two arbitrary inter-level edges with w.l.o.g. $u_1 \prec^\pi u_2$. They cross in π if and only if $v_2 \prec^\pi v_1$, whereas they cross in π' if and only if $u_2 \prec_{v_1}^{\pi'} u_1$ and $v_2 \prec_{u_1}^{\pi'} v_1$.

Now, let $e_1 = (v_1, v_3), v_2 = (v_2, v_4) \in H$ be two arbitrary intra-level edges with w.l.o.g. $v_1 \prec^\pi v_2$. They cross in π if and only if $v_2 \prec^\pi v_3 \prec^\pi v_4$, whereas they cross in π' if and only if $v_1 \prec_{v_1}^{\pi'} v_2 \prec_{v_1}^{\pi'} v_3 \prec_{v_1}^{\pi'} v_4$.

Finally, let $e_1 = (u, v_2) \in E$ be an arbitrary inter-level edge and $e_2 = (v_1, v_3) \in H$ be an arbitrary intra-level edge with w.l.o.g. $v_1 \prec^\pi v_3$. They cross in π if and only if $v_1 \prec^\pi v_2 \prec^\pi v_3$, whereas they cross in π' if and only if $v_1 \prec_u^{\pi'} v_2 \prec_u^{\pi'} v_3$ and $v_3 \prec_{v_3}^{\pi'} u \prec_{v_3}^{\pi'} v_1$. □

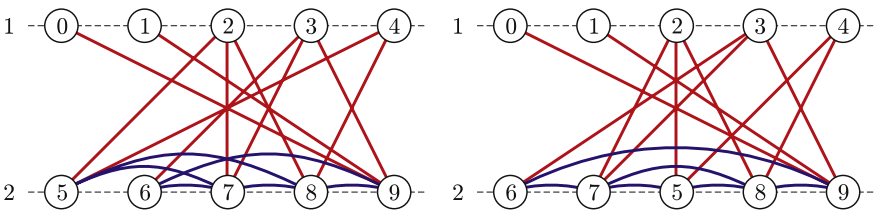
Unfortunately, this simple transformation cannot be applied to radial drawings. Thus, we present a new method for extended one-sided crossing minimization in the following section.

3.2. Extended sifting

We now present a new *extended sifting* algorithm for extended one-sided crossing minimization, where we treat the three different kinds of crossings separately without any impact on the running time and quality. The algorithm runs not only theoretically but also practically within the same time and generates exactly the same orderings and number of crossings. Further, speaking in terms of algorithm or software engineering, the problem is attacked with a more modular approach.

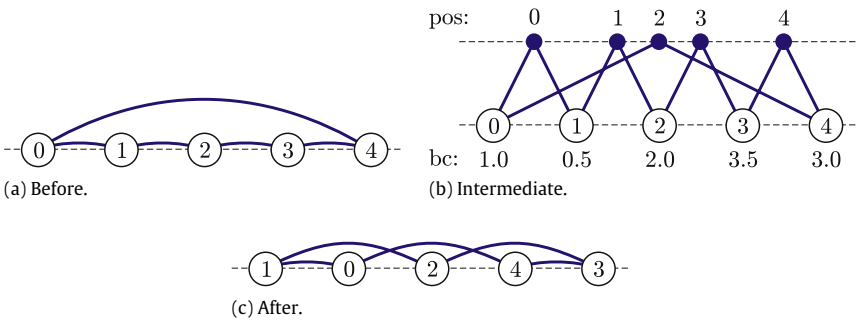
3.2.1. Intra-level sifting for crossings between intra-level edges

Consider overlapping intra-level edges $\{v_1, v_4\}, \{v_2, v_3\} \in H$ with $v_1 \prec v_2 \prec v_3 \prec v_4$. They do not cross, since we draw each edge $\{u, v\}$ as a circular arc instead of a straight line. For that, we use a quadratic spline with an amplitude, i.e., height of the only interpolation point, rising with the number of enclosed vertices between u and v in the current ordering $\prec$ of V_2. Thus, even if $v_1 = v_2$ or $v_3 = v_4$ the edges do not cross, except in common end points. We further take care not to introduce unnecessary "double" crossings between intra-level and inter-level edges by restricting the maximum edge amplitude according to the dimension of the drawing. For example, if the edge $\{5, 8\}$ in Fig. 3(a) has a higher amplitude, it would cross the edge $\{4, 5\}$. Note that we require to draw all intra-level edges completely above the second level line, as will be explained in Section 3.2.2.



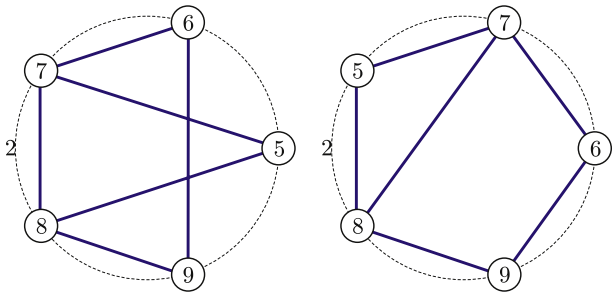
(a) Before minimization. (b) After minimization.

Fig. 3. The first two levels of Fig. 1.



(a) Before. (b) Intermediate. (c) After.

Fig. 4. Insertion Barycenter of [23].



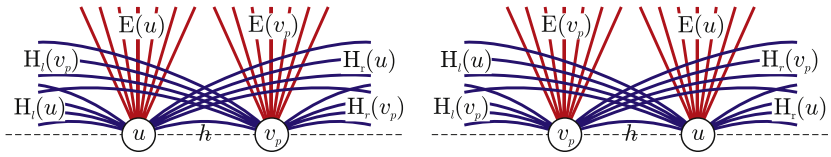
(a) Before. (b) After.

Fig. 5. Circular crossing minimization for intra-level edges of the graph in Fig. 3.

Note that Sugiyama and Misue [23] presented a faster but qualitatively inferior *insertion barycenter method* for intra-level crossing minimization. More specifically, they created a dummy vertex splitting each intra-level edge, which they placed on a common dummy level. After computing the barycenter value of the neighbors for each dummy vertex, they ordered the dummy level in ascending value. Finally, they computed the barycenter values for the original vertices according to the new positions of their dummy neighbors, which define the final ordering. Unfortunately, this method introduces unnecessary crossings, as Fig. 4 shows. A straightforward solution to avoid these unnecessary crossings may be to make the amplitudes of the edges pairwise different, which leads up to $|H|$ different dummy levels. As a consequence, in order to be able to run a 2-level crossing minimization algorithm considering all types of crossings later, each inter-level edge must be split in $|H|+1$ segments by $|H|$ new dummy vertices. This prevents not only time efficient processing, but also is obstructive for a good result, i.e., fewer crossings: Each of the additionally necessary $|H|$ crossing minimization rounds is a heuristic only and is thus not exact.

Thus, we again use the idea from circular sifting (Section 2.3), which we already have used in inter-level sifting in Section 2.4, however, now for crossing minimization between intra-level edges. Hence, we call it *intra-level sifting*.

Considering the horizontal line of level 2 bent to a circle (see Fig. 5), the circular crossing minimization algorithm fits out of the box: For one round call Algorithm 1 where line 9 of Algorithm 2 is changed to call Algorithm 4 instead of Algorithm 3. Line 3 of Algorithm 2 is left away in this case. Algorithm 4 is the same as Algorithm 3 except that the neighbors are on level 2 and the ordering $\prec$ is replaced by $\prec_{v_p}$, i.e., the ordering of V_2 is different in each swap.



(a) Before swap. (b) After swap.

Fig. 6. Crossings among intra-level and inter-level edges.

Algorithm 4. SIFTING-SWAP-INTRA

Input: Ordered 2-level graph $G=(V_1 \dot\cup V_2, E, H, \phi)$, Swap vertices $u, v_p \in V_2$
Output: Change in crossing count

```
1  let x_0 ≺_{v_p} ··· ≺_{v_p} x_{r-1} be the neighbors of u in V_2 − {v_p}, {x_i, u} ∈ H
2  let y_0 ≺_{v_p} ··· ≺_{v_p} y_{s-1} be the neighbors of v_p in V_2 − {u}, {y_j, v_p} ∈ H
3  c ← 0; i ← 0; j ← 0
4  while i < r and j < s do
5      if x_i ≺_{v_p} y_j then
6          c ← c − (s − j)
7          i ← i + 1
8      else if y_j ≺_{v_p} x_i then
9          c ← c + (r − i)
10         j ← j + 1
11     else
12         c ← c − (s − j) + (r − i)
13         i ← i + 1; j ← j + 1
14 return c
```

With Algorithm 5 we keep the ordered intra-level adjacencies of vertex u up to date during a sifting step. Thus, we know the ordering $\prec_{v_p}$ among u's neighbors, since this is the concatenation of $H_r(u)$ and $H_l(u)$ (in this order). Therefore, we need no reordering for determining the x_is per swap. The same holds for the y_is: Algorithm 5 also updates the intra-level adjacencies of the swap vertex v_p, but does not maintain their ordering due to performance restrictions, in contrast to u. However, we rely on the fact that a *short edge* $h = \{u, v_p\}$ is always the first of $H_l(v_p)$. This is true since we build up the sorting of this adjacency list right after u was placed on the first position of V_2 in Algorithm 2 and there never were any updates to this ordering. In other words, the ordering of the intra-level adjacencies of all vertices v_p is valid throughout the complete sifting step besides obsolete positions of edges $\{u, v_p\}$. However, these exceptions are irrelevant for the determination of the orderings of the y_is, since they never contain u.

Algorithm 5. UPDATE-INTRA-ADJ

Input: Ordered 2-level graph $G=(V_1 \dot\cup V_2, E, H, \phi)$, Swap vertices $u, v_p \in V_2$
Output: Number of edges between u and v_p, Updated $H_l(u)$, $H_r(u)$, $H_l(v_p)$, and $H_r(v_p)$ as side effect

```
1  l ← 0                                   // number of short intra-level edges
2  while {u, v_p} = getFirst(H_r(u)) ∈ H do
3      h ← removeFirst(H_r(u))
4      append(H_l(u), h)
5      removeFirst(H_l(v_p))               // first, since list was never updated before
6      prepend(H_r(v_p), h)
7      l ← l + 1
8  return l
```

3.2.2. *Mixed sifting for crossings between inter-level and intra-level edges*

As mentioned previously, we restrict intra-level edges to be only routed above the second level line. Otherwise, if we allowed routing on both sides, the number of crossings between inter-level and intra-level edges would depend on the inter-level edges to vertices on a third level, which contradicts the pairwise level by a level sweep approach.

As with the previous inter-level and intra-level sifting algorithms, swapping vertex u with its successor v_p changes only the crossings (here among inter-level and intra-level edges) between edges incident to u or v_p. Thus for computing the change in the crossing count, we only need the sizes of the six sets $H_l(v), H_r(v), E(v)$ with $v \in \{u, v_p\}$, see Fig. 6.

Neglecting potentially existing short edges $h = \{u, v_p\} \in H$ which is a non-contributing special case, we obtain (1) as change in crossing count Δ when swapping u and successor v_p. The correctness follows again from the invariant that after a swap exactly those pairs of intra-level (excluding short edges) and inter-level edges cross which did not cross previously.

$$\Delta = (|H_r(v_p)| - |H_l(v_p)|) \cdot |E(u)| + (|H_l(u)| - |H_r(u)|) \cdot |E(v_p)|. \tag{1}$$

Thus, for *mixed sifting* a complete round can be started by calling Algorithm 1 and updating line 9 of Algorithm 2 to call Algorithm 6. Line 3 of Algorithm 2 does not need to be executed here. Note that the intra-level adjacency updates caused by a swap are done prior to a call of Algorithm 6. Thus l has now to be subtracted from $H_l(u)$ and $H_r(v_p)$ instead of $H_r(u)$ and $H_l(v_p)$.

Algorithm 6. SIFTING-SWAP-MIXED

Input: Ordered 2-level graph $G = (V_1 \dot\cup V_2, E, H, \phi)$, Swap vertices $u, v_p \in V_2$, Number of short edges $l = |\{\{u, v_p\} \in H\}|$

Output: Change in crossing count

1 **return** $((|H_r(v_p)| - l) - |H_l(v_p)|) \cdot |E(u)| + ((|H_l(u)| - l) - |H_r(u)|) \cdot |E(v_p)|$

3.3. Combining all crossings

Finally, for our main algorithm *extended sifting* considering all types of crossings, we call Algorithm 1 with an updated line 9 of Algorithm 2 in order to call Algorithm 7. There, we simply add the three independent changes in crossing counts. However, other formulas preferring some type of crossings at the expense of more crossings of other types are possible, e.g., the usage of weighting factors.

Algorithm 7. SIFTING-SWAP-EXT

Input: Ordered 2-level graph $G = (V_1 \dot\cup V_2, E, H, \phi)$, Swap vertices $u, v_p \in V_2$

Output: Change in crossing count

1 $c_E \leftarrow$ SIFTING-SWAP-INTER(G, u, v_p)
2 $c_H \leftarrow$ SIFTING-SWAP-INTRA(G, u, v_p)
3 $c_{HE} \leftarrow$ SIFTING-SWAP-MIXED(G, u, v_p)
4 **return** $c_V + c_H + c_{HV}$

We obtain the same time bound as the original sifting algorithm for a level graph $G = (V, E, \phi)$ considering only inter-level edges, or for a graph $G = (V, H)$ considering only intra-level edges.

Theorem 3.1. *One round of extended one-sided sifting on an extended 2-level graph $G = (V, E, H, \phi)$ takes $\mathcal{O}(|V| \cdot (|E| + |H|))$ time.*

Proof. For running time calculations, we assume w. l. o. g. that there are no isolated vertices. They can be removed in a preprocessing step and added again in postprocessing since their positions have no influence on the crossing number.

One round of inter-level (intra-level) sifting takes $\mathcal{O}(|V| \cdot |E|)$ ($\mathcal{O}(|V| \cdot |H|)$) time according to Theorem 3 of [4]. One round of mixed sifting takes $\mathcal{O}(|V| \cdot |H|)$ time, since one step needs $\mathcal{O}(|H|)$ time: The initial sorting of the intra-level adjacency in Algorithm 2 can be done in $\mathcal{O}(|H|)$ time by traversing the vertices of V_2 in order and adding each to the adjacency lists of its right or left neighbors. Each of the $|V_2|$ sifting swaps takes constant time. An integrated execution is possible, since the only updates to the intra-level adjacency list are done by Algorithm 2. Thus the algorithms mutually do not compromise each other. □

3.4. Sweep over all Levels

According to our experience, the quality of sifting does not depend much on the quality of the initial vertex ordering of the first level. However, a "bad" initialization raises the number of needed sifting rounds and thus the absolute running time. Therefore, it may be useful to apply some rounds of intra-level sifting to V_1 to get a practical initial ordering.

In a top-down sweep, we reorder the levels i from 2 to k by consecutively applying our extended one-sided 2-level crossing minimization on the fix ordered set V_{i-1} and on the freely permutable set V_i. In the subsequent bottom-up sweep we reorder the levels i from $k-1$ down to 1 by consecutively applying the extended one-sided 2-level crossing minimization on the fix ordered set V_{i+1} and the permutable set V_i. However, in the bottom-up sweep we have a slightly different situation, since the intra-level edges are below the current level i and cross edges from level i and $i-1$ (see Fig. 7 for an example).

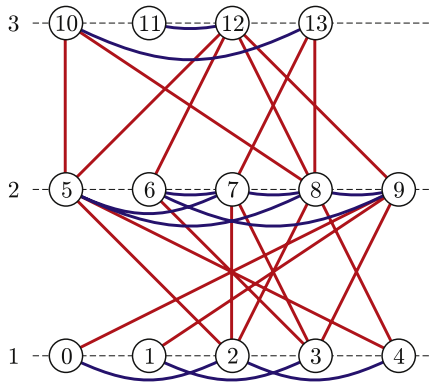


Fig. 7. Bottom-up sweep for the graph in Fig. 1.

Nevertheless, the formula for crossings of intra-level and inter-level edges (1) does not depend on any vertex ordering different to that on level i and especially does not depend on that of level $i-1$. Thus, we count the change in the number of crossings of the intra-level edges of level i with the inter-level edges between level i and $i-1$ during a swap. For this, we let for every $v \in V_i$ be $E(v) = \{(x, v) \mid x \in V_{i-1}\}$ instead of $\{(y, v) \mid y \in V_{i+1}\}$ in Algorithm 6, which then is the same as a top-down sweep. After some iterations (for our experiments 10) of top-down with subsequent bottom-up sweeps the algorithm terminates.

4. Crossing minimization on concentric levels

In this section, we present a heuristic for extended one-sided crossing minimization with *radial drawings*. In radial drawings, we place vertices on concentric circles, instead of parallel horizontal lines [1,2]. The major advantage of radial over horizontal drawings is the additional freedom of routing edges in two directions around the center, i.e., clockwise and counterclockwise, which results in fewer edge crossings and therefore reduced visual complexity. Further, there is also a higher probability of no crossings, because of the fact that the set of level planar graphs is a proper subset of radial planar graphs [2].

As in the previous section, to display extended k-level graphs $G = (V, E, H, \phi)$, we define *extended radial drawings* as radial drawings with additional intra-level edges. For crossing minimization in extended radial drawings, we use exactly the same framework as for horizontal drawings, described in the previous section. Thus, we restrict ourselves to the only difference, the *one-sided radial 2-level crossing minimization*, in the following.

4.1. Radial sifting for crossing minimization in radial drawings

In this section we briefly review the results and terminology of *radial sifting* [1] for crossing minimization in radial drawings of level graphs $G = (V_1 \dot\cup V_2, E, \phi)$ containing inter-level edges, an adaption of the original sifting algorithm to radial drawings.

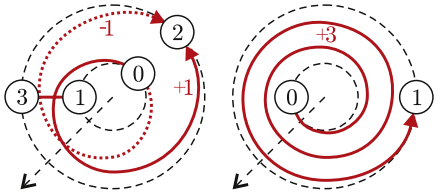
In order to represent orderings π_1 and π_2 (w.l.o.g. counterclockwise) of the vertices on the circular levels, a *ray* is introduced as a straight half-line from the concentric center to infinity which tags the borderline between the vertices with extremal positions. Edges crossing the ray are called *cut edges*.

How many times and in which *direction* an edge is wound around the center is crucial for radial drawings. This information is stored by the *offset* $\psi : E \to \mathbb{Z}$ 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* (*counterclockwise*) cut edge, i.e., the sign of $\psi(e)$ reflects the mathematical direction of rotation, see Fig. 8. If $\psi(e) = 0$, then e is no cut edge and thus needs no direction information. Observe that a cut edge cannot cross the ray clockwise and counterclockwise simultaneously and for a small number of crossings only offsets in $\{-1, 0, 1\}$ are of interest. A *radial embedding* $\mathcal{E}$ of G is defined by the vertex order π and the edge offsets ψ, i.e., $\mathcal{E} = (\pi, \psi)$.

Lemma 4.1 (*[1]*). *Radial one-sided 2-level crossing minimization is $\mathcal{NP}$-hard.*

For radial drawings, however, the idea of circular sifting cannot be adopted directly, as the crossing number between inter-level edges also depends on their offsets, which are not necessarily constant. Thus inter-level sifting as described in Section 2.3 is used, with a different formula for counting crossings ($\mathrm{sgn} : \mathbb{R} \to \{-1, 0, 1\}$ is the signum function):

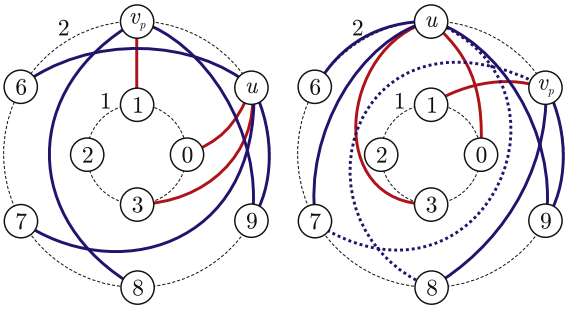
Lemma 4.2 (*[1]*). *Let $\mathcal{E} = (\pi, \psi)$ be a radial embedding of a 2-level graph $G = (V_1 \dot\cup V_2, E, \phi)$. Then the number of crossings between two edges $e_1 = (u_1, v_1) \in E$ and $e_2 = (u_2, v_2) \in E$ is*



(a) Edge (0, 2) drawn counterclockwise and clockwise (dotted).

(b) $\psi((0,1)) = +3$.

Fig. 8. Offsets of edges [1].



(a) Compute number of crossings before swap.

(b) Update number of crossings after swap.

Fig. 9. Extended radial sifting: crossings between inter-level and intra-level edges and optimal routing.

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

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

Before counting the change in crossing number by considering the edges incident to those two swapped vertices, it is crucial to adapt the offsets of the edges. Let u be the vertex moved along in counterclockwise direction in a sifting step. Initially, the offset of all edges $(\cdot, u)$ are set to 1 and ordered according to the positions of incident vertices on level 1. While moving u along level 2 in counterclockwise direction, the offsets of edges are decreased according to their ordering by 1 as long as that reduces the number of crossings. The split in the edge list is called the *parting*, i.e., where the offsets differ by 1. The parting may move around the center twice, as offsets can be decreased from 1 to -1.

Theorem 4.1 (*[1]*). *Given a 2-level graph* $G = (V, E, \phi)$, *radial sifting runs in* $\mathcal{O}(|V|^2 \cdot |E|)$ *time.*

4.2. Intra-level edges in extended radial drawings

We now discuss the new problem of *extended radial crossing minimization* for extended 2-level graphs $G = (V_1 \dot\cup V_2, E, H, \phi)$. For an *extended radial drawing* we draw intra-level edges as segments of circles with different radii (with the interpolation point in the middle) according to the edge lengths, i.e., the number of spanned vertices plus 1. The interpolation point of longer edges is closer to the concentric center, see Fig. 1(b) for an example. Intra-level edges lie in the inner face of level 2, but must not intersect with the circle of level 1. Further, they must be "flat" enough, i.e., they must nestle to the circle of level 2 close enough, not intersecting inter-level edges of their end vertices unnecessarily. Contrary to straight line edges, there are two possibilities to wind the edges around the center, clockwise or counterclockwise. For a low crossing number and low visual complexity, we always use the direction with shorter length, i.e., $\leq \lfloor \frac{|V_2|}{2} \rfloor$. See Fig. 9(b) where *long edges* are marked by dotted lines. For an easy notation $(v_1, v_2) \in H$ denotes an intra-level edge that is wound counterclockwise around the center starting at vertex v_1 and ending at vertex v_2. This partitions the intra-level adjacency $H(v)$ for each $v \in V_2$ in two sets, the *incoming set* $H_l(v) = \{h = (\cdot, v) \mid h \in H\}$ and the *outgoing set* $H_r(v) = \{h = (v, \cdot) \mid h \in H\}$. Per convention, we store edges for which each direction results in the same length in H_l. H_l and H_r are again kept sorted similar to Algorithm 5. As a consequence, both the left and the right adjacency lists of both vertices are updated after their sifting swap.

4.3. Extended radial sifting for crossing minimization in extended level graphs

In this section, we present our *extended radial sifting* heuristic for the extended one-sided radial crossing minimization problem. We first analyze the time complexity of the problem.

Lemma 4.3. *Extended radial one-sided 2-level crossing minimization is $\mathcal{NP}$-hard.*

Proof. It is straightforward that the extended one-sided radial crossing minimization is $\mathcal{NP}$-hard, as the two subproblems, circular crossing minimization [18] and the radial one-sided 2-level crossing minimization [1] are $\mathcal{NP}$-hard. □

For extended radial sifting we also use the same modular approach as in extended intra-level sifting described in Section 3.2 to reduce the three different types of crossings, and add the resulting changes using the separate crossing numbers. For the minimization of crossings only between inter-level edges, the radial sifting heuristic is used as it is. Also, drawing intra-level edges as segments of circles instead of straight-lines clearly does not change the number of crossings between them. Thus, we can use the circular sifting heuristic. However, the algorithm for crossing minimization of crossings between intra-level edges and inter-level edges needs some modification, as will be described in the following section.

4.3.1. Crossings between inter-level and intra-level edges in extended radial drawings

Since we have split the intra-level adjacencies in incoming and outgoing edges, computing the change in crossings when swapping two consecutive vertices u and v_p stays principally the same as in (1). Thereby, we again neglect short edges $\{u, v_p\}$ which do not contribute to the crossings. What remains is the additional freedom of routing the intra-level edges around the center in two different directions. Contrary to crossings between only intra-level edges, this now has an effect, see Fig. 9 for an example. To overcome this problem, we use the heuristic to always prefer the shorter direction.

We denote intra-level edges that span at least $\lfloor \frac{|V_2|}{2} \rfloor - 1$ vertices as *long edges*. After the swap, the length of all incoming intra-level edges of u, in $H_l(u)$ and all outgoing intra-level edges of v_p in $H_r(v_p)$ is increased by 1. Likewise, the length of all outgoing edges of u in $H_r(u)$ and all incoming edges of v_p in $H_l(v_p)$ is decreased by 1. In the case of an increase, it only can happen that the first (last) edge of the adjacency list $H_l(u)$ ($H_r(v_p)$) becomes a long edge. This is true, since we keep the adjacency lists ordered according to $\prec_u$ ($\prec_{v_p}$) and ascending edge lengths. The necessary updates are done with Algorithm 8. The number of crossings with inter-level edges COUNT-MIXED-CROSSINGS(h) caused by an intra-level edge $h = (v_1, v_2)$ is the number of inter-level edges which have exactly one incident vertex between v_1 and v_2 and the other one outside.

Algorithm 8. SHORTEN-LONG-EDGES

Input: Ordered 2-level graph $G = (V_1 \dot\cup V_2, E, H, \phi)$, Swapped vertices $v_p, u \in V_2$
Output: Change in crossing count

```
1  c ← 0
2  while h = (o, u) = getFirst(H_l(u)) ∈ H is long do
3      c ← c − COUNT-MIXED-CROSSINGS(h)
4      removeFirst(H_l(u))
5      remove(H_r(o), h)
6      h ← (u, o)                          // swap direction
7      append(H_r(u), h)
8      prepend(H_l(o), h)
9      c ← c + COUNT-MIXED-CROSSINGS(h)
10 while h = (v_k, o) = getLast(H_r(v_p)) ∈ H is long do
11     c ← c − COUNT-MIXED-CROSSINGS(h)
12     removeLast(H_r(v_p))
13     remove(H_l(o), h)
14     h ← (o, v_k)                        // swap direction
15     prepend(H_l(v_p), h)
16     append(H_r(o), h)
17     c ← c + COUNT-MIXED-CROSSINGS(h)
18 return c
```

One special case remains: If $|V_2|$ is even, then some intra-level edges may have the same length $\frac{|V|}{2}$ in both directions. We call them *vis-à-vis edges*, since they are incident to two vertices that are placed opposite to each other. In order to locally minimize the number of crossings, we break ties in favor of the direction that causes less mixed crossings as shown in Algorithm 9. We update the adjacency lists according to which direction of the current edge causes less crossings.

Algorithm 9. SHORTEN-VISAVIS-EDGES

Input: Ordered 2-level graph $G=(V_1 \dot\cup V_2, E, H, \phi)$, Swapped vertices $v_p, u \in V_2$
Output: Change in crossing count

```
1  c ← 0
2  while h = (o, u) = getFirst(H_l(u)) ∈ H is vis-à-vis do
3      c_1 ← COUNT-MIXED-CROSSINGS(h)
4      removeFirst(H_l(u))
5      remove(H_r(o), h)
6      h ← (u, o)                          // swap direction
7      append(H_r(u), h)
8      prepend(H_l(o), h)
9      c_2 ← COUNT-MIXED-CROSSINGS(h)
10     if c_2 ≤ c_1 then c ← c + c_2 − c_1
11     else undo changes
12 while h = (v_k, o) = getLast(H_r(v_p)) ∈ H is vis-à-vis do
13     c_1 ← COUNT-MIXED-CROSSINGS(h)
14     removeLast(H_r(v_p))
15     remove(H_l(o), h)
16     h ← (o, v_k)                        // swap direction
17     prepend(H_l(v_p), h)
18     append(H_r(o), h)
19     c_2 ← COUNT-MIXED-CROSSINGS(h)
20     if c_2 ≤ c_1 then c ← c + c_2 − c_1
21     else undo changes
22 return c
```

4.3.2. All crossings in extended radial drawings

The overall sifting swap is essentially analogous to Algorithm 7. First we compute the change in the number of crossings between inter-level edges, then between intra-level edges, and finally between intra-level and inter-level edges. However, the sifting step (Algorithm 10) is extended as more updating parts are needed: After the preliminary steps and swapping the current vertex with its successor in the sifting swap, the offsets and parting of the involved inter-level edges must be updated as well as the routing of some intra-level edges, depending on their length.

Algorithm 10. SIFTING-STEP

Input: Ordered 2-level offset graph $G=(V_1 \dot\cup V_2, E, H, \phi_E)$, Vertex $u \in V_2$ to sift
Output: Updated ordering of V_2

```
1  let v_0 = u ≺ v_1 ≺ ··· ≺ v_{|V_2|−1} be the current ordering of V_2 with u put to front
2  sort all edges
3  find best parting and offsets for edges (·, u) ∈ E
4  χ ← 0; χ* ← 0                          // current and best number of crossings
5  p* ← 0                                 // best vertex position
6  j ← 0; j* ← 0                          // current and best offset at the parting
7  i ← 0; i* ← 0                          // current and best parting
8  for p ← 1 to |V_2| − 1 do
9      χ ← χ+SIFTING-SWAP-EXT(G, u, v_p)
10     χ ← χ+SHORTEN-LONG-EDGES(G, u, v_p)
11     χ ← χ+UPDATE-OFFSETS(G, u, v_p)
12     χ ← χ+SHORTEN-VISAVIS-EDGES(G, u, v_p)
13     if χ < χ* then
14         χ* ← χ; p* ← p
15         j* ← j
16         i* ← i
17 set best parting and offsets for edges (·, u) ∈ E
18 return V_2 ← v_1 ≺ ··· ≺ v_{p*−1} ≺ u ≺ v_{p*} ≺ ··· ≺ v_{|V_2|−1}
```

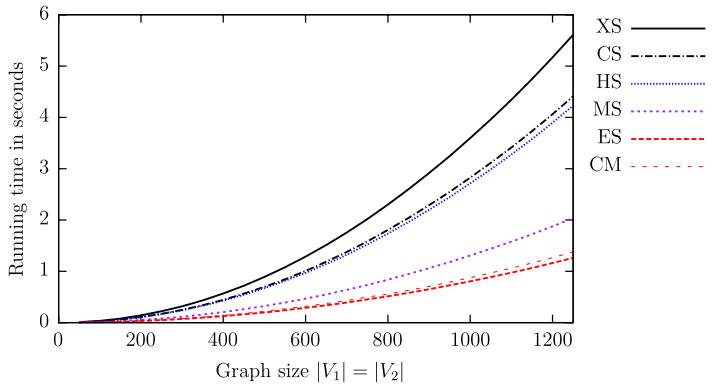


Fig. A.1. Benchmark: running times.

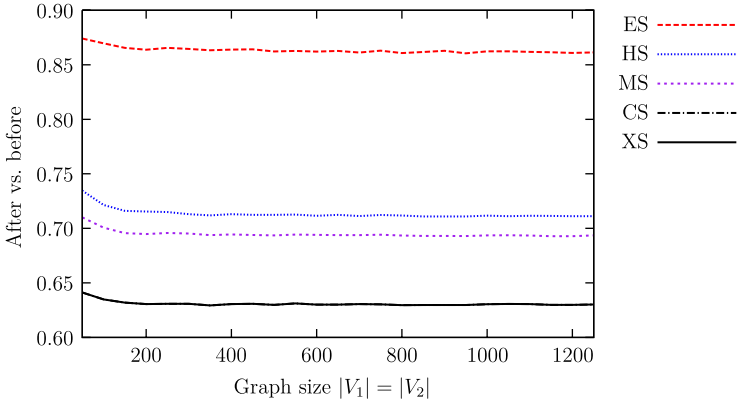


Fig. A.2. Benchmark: total crossing numbers.

4.3.3. Computational complexity

For our extended radial sifting, we use the radial sifting algorithm as a subcomponent. Thereby, we consider the numbers of intra-level and mixed crossings additionally to the inter-level crossings. We obtain the following time complexity:

Theorem 4.2. *One round of extended radial sifting on an extended 2-level graph* $G = (V, E, H, \phi)$ *takes* $\mathcal{O}(|V|^2 \cdot (|E| + |H|))$ *time.*

Proof. The worst case in terms of computational complexity occurs for the current vertex u being a vertex with inter-level and intra-level edges. In that case, the running time for one round of radial sifting is $\mathcal{O}(|V_2|^2 \cdot |E| + |E|^2)$ and for circular sifting $\mathcal{O}(|V_2| \cdot |H|)$.

Before and after a sifting swap, the computation and update of the mixed crossing number between two consecutive vertices u and v_p without any intra-level edges involved that change their direction runs in $\mathcal{O}(1)$ time. As in one round of sifting each vertex is at each position once, this contributes $\mathcal{O}(|V_2|^2)$ to the overall running time. If an intra-level edge $h \in H$ changes direction, the computation of its number of crossings with inter-level edges runs in $\mathcal{O}(|E|)$ time. Now consider only one sifting step, i.e., one vertex u is moved along the periphery of its level. Let $h = (v_1, v_2)$ be an edge incident to two vertices $v_1, v_2 \in V_2$ with $v_1, v_2 \neq u$. Both v_1 and v_2 only change their position once during the sifting step of u. Thus, h can change its direction at most twice in one sifting step. Considering the incident intra-level edges of u, they can change their direction at most twice as well. Therefore, the contribution to the time complexity for one sifting step is $\mathcal{O}(|V_2| \cdot |E| + |H|)$ and thus for one round of sifting is $\mathcal{O}(|V_2|^2 \cdot |E|)$. □

5. Experimental results

To analyze the performance of one sifting round of our extended one-sided 2-level crossing minimization heuristics, we have implemented them in Java. Further, we have implemented the corresponding standard sifting algorithm which uses a crossing matrix to compare its practical running time with the sifting algorithm of [4]. We have evaluated the implementations for horizontal drawings using 15 625 random level graphs: 25 graphs for each combination of the parameters $|V_1| = |V_2| \in \{50, 100, \ldots, 1250\}$, $|E|/|V_2| \in \{1, \ldots, 5\}$, and $|H|/|V_2| \in \{1, \ldots, 5\}$. With the same parameters, but $|V_1| = |V_2| \in \{20, 30, \ldots, 290\}$, we similarly have tested 17 500 random radial level graphs.

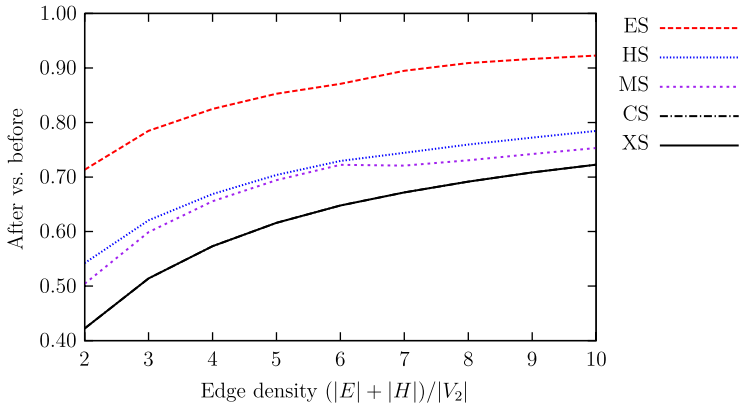


Fig. A.3. Benchmark: total crossing numbers.

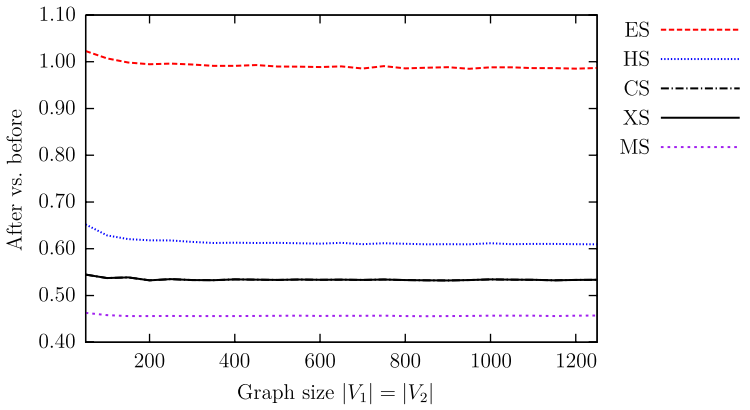


Fig. A.4. Benchmark: numbers of crossings between intra-level and inter-level edges.

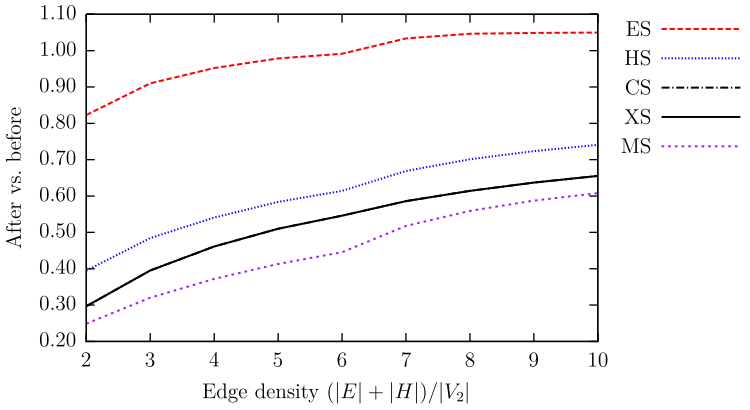


Fig. A.5. Benchmark: numbers of crossings between intra-level and inter-level edges.

Figs. A.2 and A.3 (Figs. A.11 and A.12 for concentric levels) confirm that it makes sense to consider all types of crossings simultaneously, since the algorithms generate (as expected) fewer crossings than standard sifting, experimentally by a factor of 0.7. This is a very encouraging result, since the differences in absolute running times between our extended sifting and the existing standard (inter-level) sifting and intra-level sifting, i.e., the running time of mixed sifting, are negligible in practice even on larger graphs (see Figs. A.1 and A.10). For example, in our experiment the running time of extended sifting for horizontal drawings with $|V_1| = |V_1| = |E| = |H| = 10^4$ is about 4 minutes and for radial drawings with $|V_1| = |V_1| = |E| = |H| = 10^3$ is about 6 minutes.

To give a feeling about the performance of inter-level sifting in horizontal drawings compared to an optimal algorithm: The ILP approach of [16] using the free lpsolve library needs for $|V_1| = |V_2| = 150$ and $|E| = 750$ about 50 min to reduce the

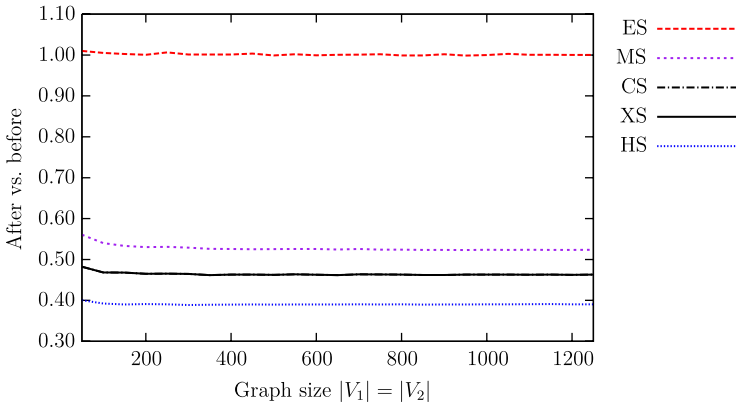


Fig. A.6. Benchmark: numbers of crossings between intra-level edges.

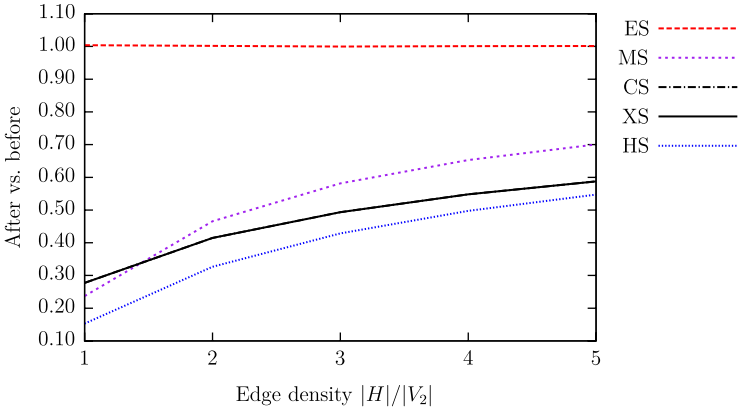


Fig. A.7. Benchmark: numbers of crossings between intra-level edges.

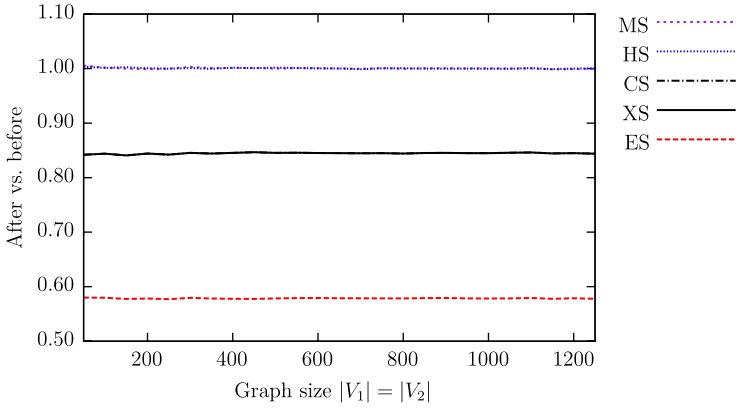


Fig. A.8. Benchmark: numbers of crossings between inter-level edges.

number of crossings from 145 925 to the optimum of 94 742. One round of sifting needs less than 20 ms and leaves 94 981 crossings. After 3 rounds we have 94 770 crossings.

6. Conclusion

In this paper, we studied the new problem of crossing minimization in extended level graphs. We considered two different drawings, horizontal drawing and radial drawings, and presented two heuristics for extended one-sided crossing minimization in both drawings. Essentially, we extended the well-known sifting heuristic for crossing minimization of level graphs to handle three different types of crossings in extended level graphs. Ignoring non-contributing self loops, our algorithms can work also on multi-graphs within the same time bounds.

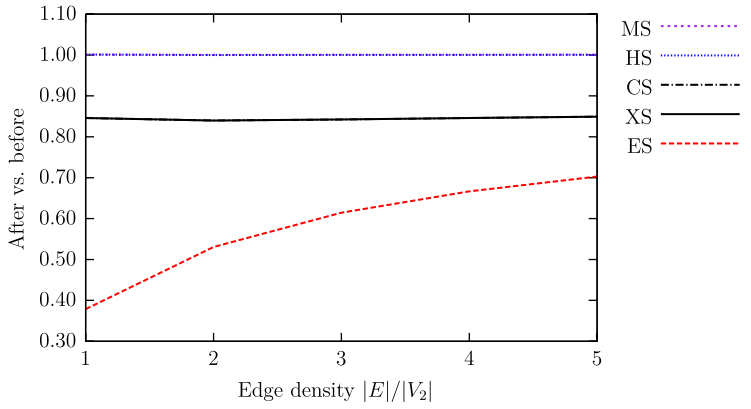


Fig. A.9. Benchmark: numbers of crossings between inter-level edges.

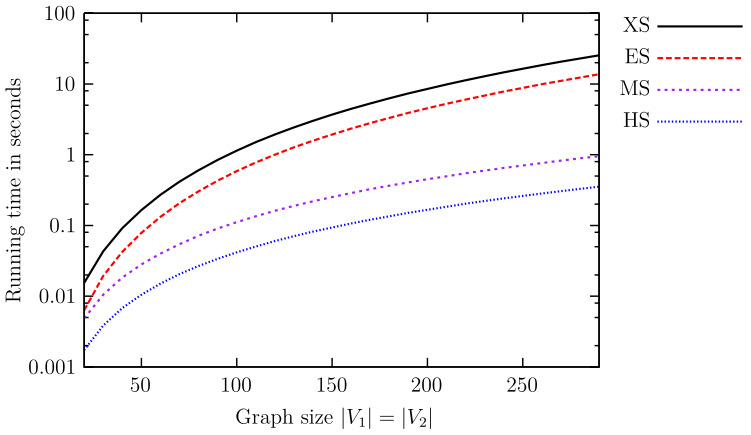


Fig. A.10. Benchmark: running times.

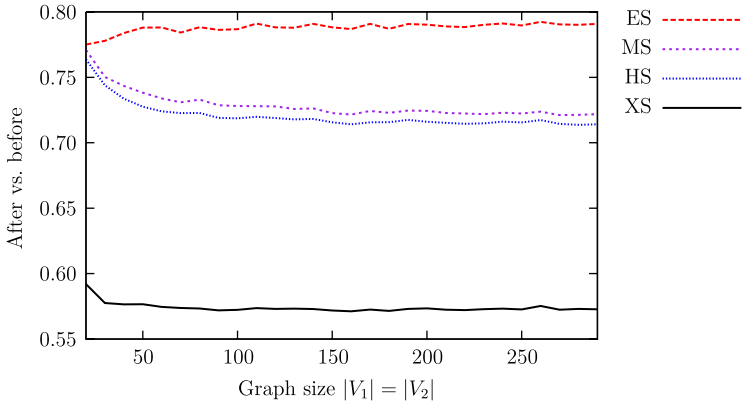


Fig. A.11. Benchmark: total crossing numbers.

So far, we have used only random initial orderings of the vertices. However, the quality of the orderings produced by extended sifting is not independent from the input. Thus, it may be helpful to use some extensions of fast and simple heuristics, e.g., barycenter or median [1,17] heuristics, to reduce crossings in a preprocessing step.

One future research is the investigation on the freedom of routing intra-level edges above and below the level lines, not restricting them to one side. As an alternative for the crossing minimization approach, a *planarization* approach was also studied for extended level graphs [10]: The planarization problem of extended-level graph is $\mathcal{NP}$-hard and, thus, heuristics are suggested. However, these should be evaluated with extensive experimental results.

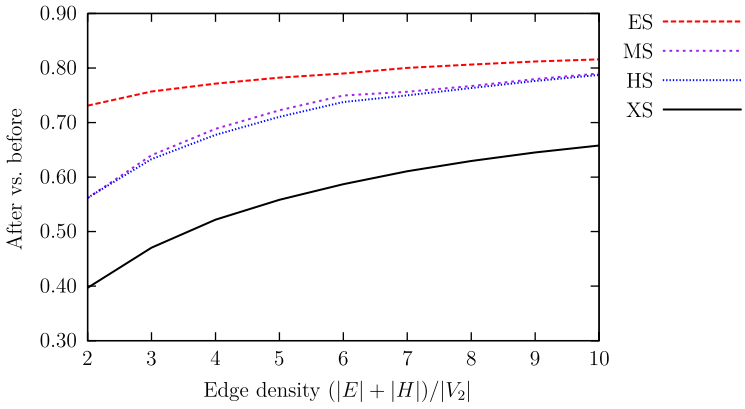


Fig. A.12. Benchmark: total crossing numbers.

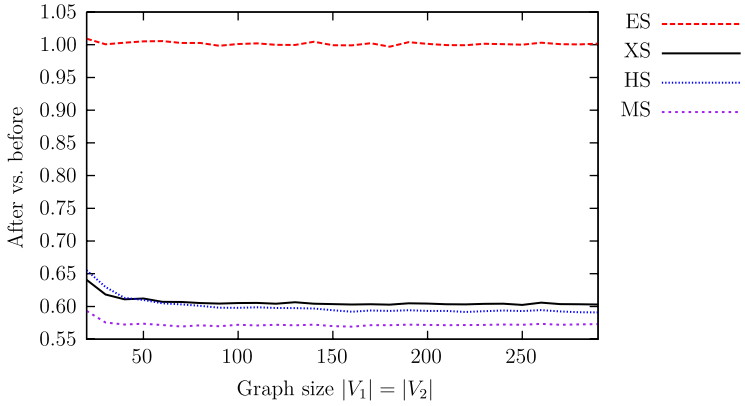


Fig. A.13. Benchmark: numbers of crossings between intra-level and inter-level edges.

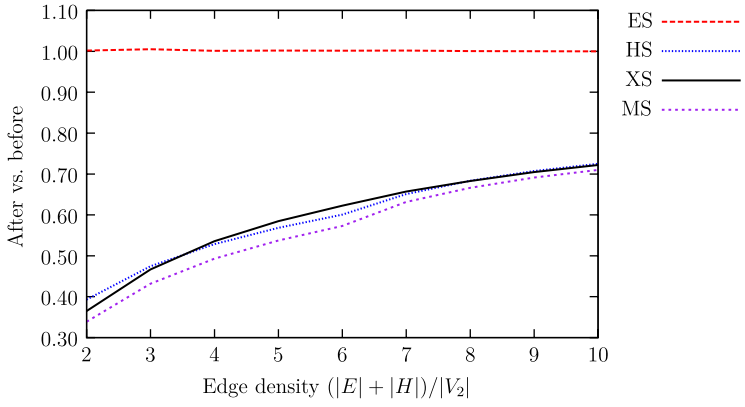


Fig. A.14. Benchmark: numbers of crossings between intra-level and inter-level edges.

Appendix A. Benchmark results

The appendix shows the practical performance of the algorithms. All benchmarks were run on a 2.4 GHz Core 2 PC under the Java 6.0 platform from Sun Microsystems, Inc.

A.1. Horizontal level lines

Figs. A.1–A.9 provide benchmark results comparing the heuristics to minimize crossings in horizontal drawings: inter-level sifting (CM with crossing matrix as in Section 2.1, ES without as in Section 2.2), circular sifting with semi-circles (CS)

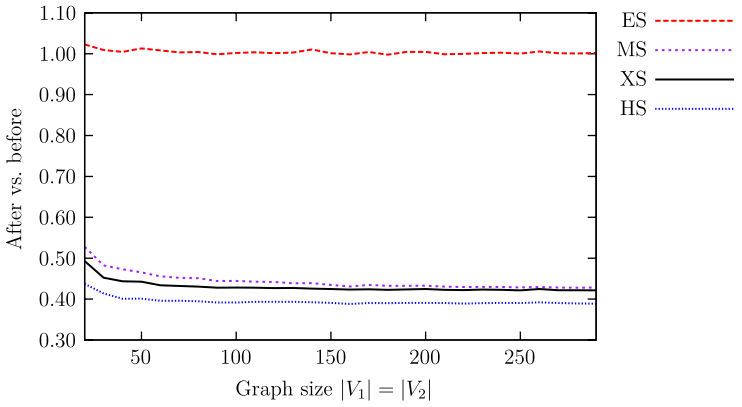


Fig. A.15. Benchmark: numbers of crossings between intra-level edges.

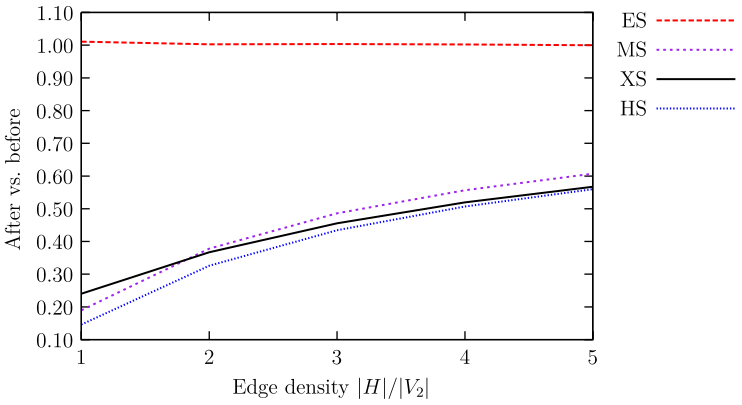


Fig. A.16. Benchmark: numbers of crossings between intra-level edges.

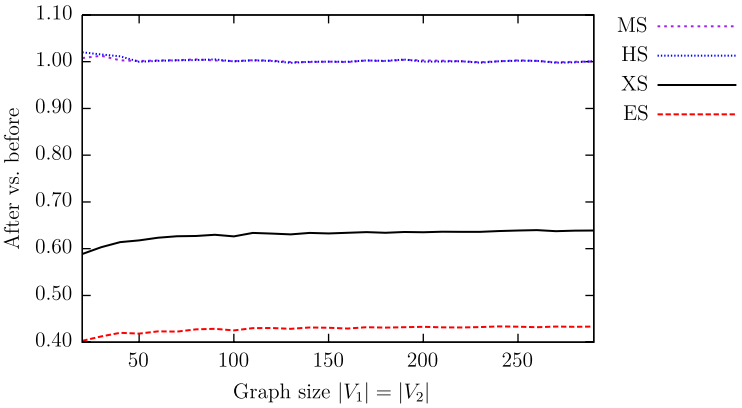


Fig. A.17. Benchmark: numbers of crossings between inter-level edges.

as in Section 3.1, intra-level sifting (HS) as in Section 3.2.1, mixed sifting (MS) as in Section 3.2.2, and extended sifting (XS) as in Section 3.3.

A.2. Concentric level lines

Figs. A.10–A.18 provide benchmark results comparing the heuristics to minimize crossings in radial drawings: inter-level sifting (ES) as in Section 4.1, intra-level sifting (HS) as in Section 4.2, mixed sifting (MS) as in Section 4.3.1, and extended sifting (XS) as in Section 4.3.2.

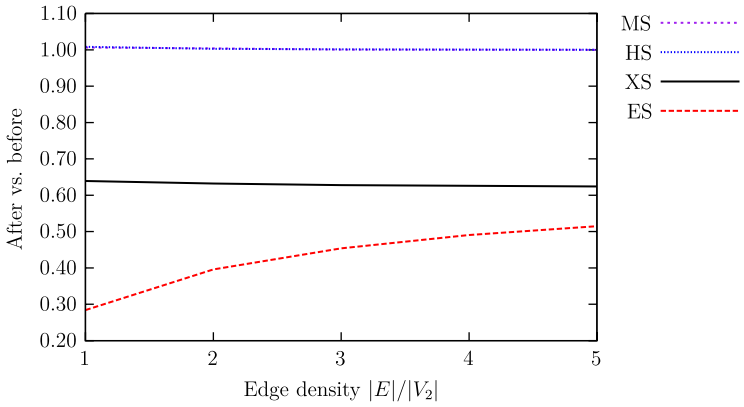
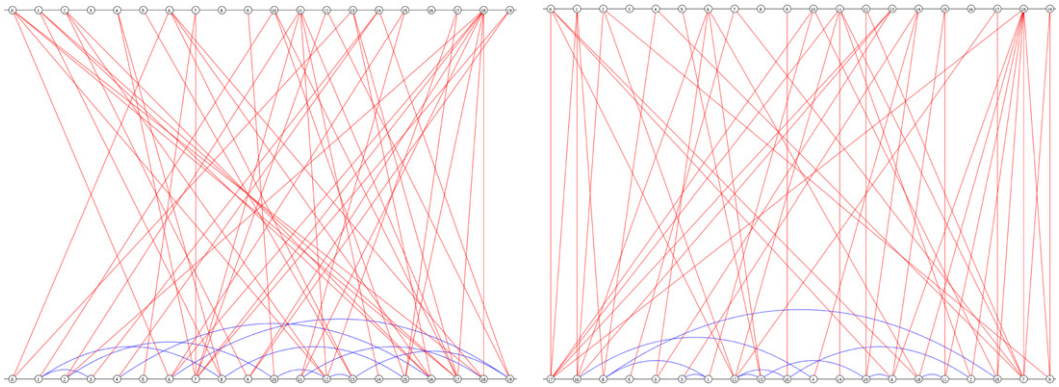


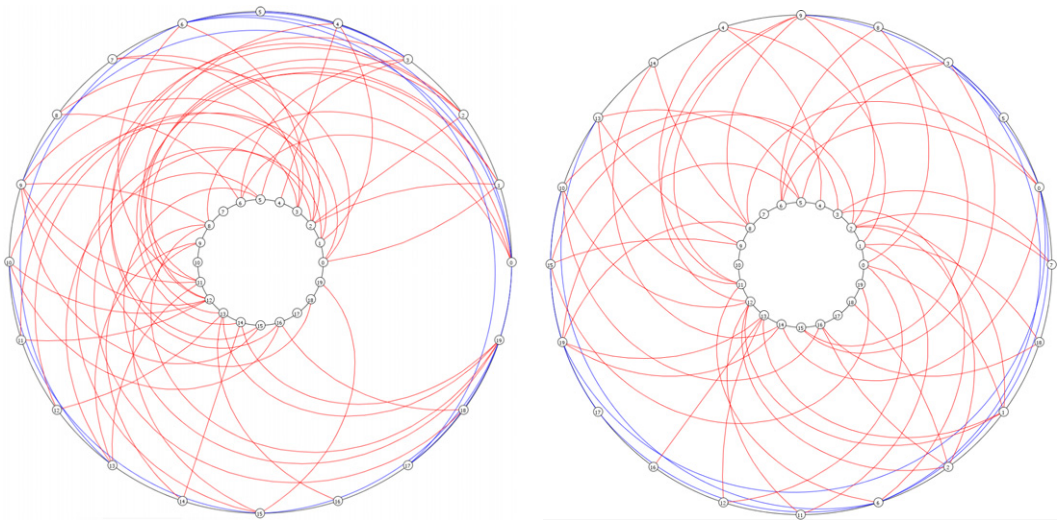
Fig. A.18. Benchmark: numbers of crossings between inter-level edges.



(a) Input with $\chi(\pi) = 768$.

(b) Output after one sifting round with $\chi(\pi') = 441$.

Fig. B.1. Horizontal example with $|V_1| = |V_2| = 20$, $|E| = 50$, and $H = 10$.



(a) Input with $\chi(\pi) = 2405$.

(b) Output one sifting round with $\chi(\pi') = 1397$.

Fig. B.2. Radial example with $|V_1| = |V_2| = 20$, $|E| = 50$, and $H = 10$.

Appendix B. Examples

Figs. B.1 and B.2 show example outputs of both extended sifting algorithms. Remember, these are not results of a complete Sugiyama framework, as the fourth phase for the coordinate assignment is missing. Here, the vertices are uniformly distributed on their levels to show the orderings.

References

[1] C. Bachmaier, A radial adaption of the sugiyama framework for visualizing hierarchical information, IEEE Transactions on Visualization and Computer Graphics 13 (3) (2007) 583–594.
[2] C. Bachmaier, F.J. Brandenburg, M. Forster, Radial level-planarity testing and embedding in linear time, Journal of Graph Algorithms and Applications 9 (1) (2005) 53–97.
[3] M. Baur, U. Brandes, Multi-circular layout of micro/macro graphs, in: S. Hong, T. Nishizeki (Eds.), Proc. Graph Drawing, GD 2007, in: LNCS, vol. 4875, Springer, 2008, pp. 255–267.
[4] M. Baur, U. Brandes, Crossing reduction in circular layout, in: J. Hromkovic, M. Nagl, B. Westfechtel (Eds.), Proc. Workshop on Graph-Theoretic Concepts in Computer Science, WG 2004, in: LNCS, vol. 3353, Springer, 2005, pp. 332–343.
[5] S. Borgatti, S. Kobourov, O. Kohlbacher, P. Mutzel, Graph drawing with applications to bioinformatics and social sciences, in: Schloss Dagstuhl seminar 08191 (Mai 2008).
[6] U. Brandes, T. Erlebach (Eds.), Network Analysis, Methodological Foundations, in: LNCS Tutorial, vol. 3418, Springer, 2005.
[7] U. Brandes, P. Kenis, D. Wagner, Centrality in policy network drawings, in: J. Kratochvíl (Ed.), Proc. Graph Drawing, GD 1999, in: LNCS, vol. 1731, Springer, 1999, pp. 250–258.
[8] U. Brandes, P. Kenis, D. Wagner, Communicating centrality in policy network drawings, IEEE Transactions on Visualization and Computer Graphics 9 (2) (2003) 241–253.
[9] U. Brandes, J. Raab, D. Wagner, Exploratory network visualization: Simultaneous display of actor status and connections, Journal of Social Structure 2 (4) (2001) 1–28.
[10] H. Buchner, Displaying centrality of a network using orbital layout, Master's Thesis, University of Passau/National ICT Sydney, 2006.
[11] G. Di Battista, P. Eades, R. Tamassia, I.G. Tollis, Graph Drawing: Algorithms for the Visualization of Graphs, Prentice Hall, 1998.
[12] P. Eades, D. Kelly, Heuristics for reducing crossings in 2-layered networks, Ars Combinatorica 21 (A) (1986) 89–98.
[13] P. Eades, N.C. Wormald, Edge crossings in drawings of bipartite graphs, Algorithmica 11 (1) (1994) 379–403.
[14] S.-H. Hong, H. Nagamochi, Approximating crossing minimization in radial layouts, in: E. Sany Laber, C.F. Bornstein, L.T. Nogueira, L. Faria (Eds.), Proc. Latin American Theoretical Informatics Symposium, LATIN 2008, in: LNCS, vol. 4957, Springer, 2008, pp. 461–472.
[15] M. Jünger, E.K. Lee, P. Mutzel, T. Odenthal, A polyhedral approach to the multi-layer crossing minimization problem, in: G. Di Battista (Ed.), GD 1997, vol. 1353, Springer, 1997, pp. 13–24.
[16] M. Jünger, P. Mutzel, 2-layer straightline crossing minimization: Performance of exact and heuristic algorithms, Journal of Graph Algorithms and Applications 1 (1) (1997) 1–25.
[17] M. Kaufmann, D. Wagner, Drawing Graphs, in: LNCS, vol. 2025, Springer, 2001.
[18] S. Masuda, T. Kashiwabara, K. Nakajima, T. Fujisawa, On the $\mathcal{NP}$-completeness of a computer network layout problem, Proc. IEEE International Symposium on Circuits and Systems (1987) 292–295.
[19] C. Matuszewski, R. Schönfeld, P. Molitor, Using sifting for k-layer straightline crossing minimization, in: J. Kratochvíl (Ed.), Proc. Graph Drawing, GD 1999, in: LNCS, vol. 1731, Springer, 1999, pp. 217–224.
[20] H. Nagamochi, An improved bound on the one-sided minimum crossing number in two-layer drawings, Discrete & Computional Geometry 33 (2005) 569–591.
[21] H.C. Purchase, Which aesthetic has the greatest effect on human understanding? in: G. Di Battista (Ed.), Proc. Graph Drawing, GD 1997, in: LNCS, vol. 1353, Springer, 1997, pp. 248–261.
[22] R. Rudell, Dynamic variable ordering for ordered binary decision diagrams, in: Proc. IEEE/ACM International Conference on Computer Aided Design, ICCAD 1993, IEEE Comp. Society Press, 1993, pp. 42–47.
[23] K. Sugiyama, K. Misue, Visualization of structural information, IEEE Transactions on Systems, Man, and Cybernetics 21 (4) (1991) 876–891.
[24] K. Sugiyama, S. Tagawa, M. Toda, Methods for visual understanding of hierarchical system structures, IEEE Transactions on Systems, Man, and Cybernetics 11 (2) (1981) 109–125.
[25] V. Valls, R. Marti, P. Lino, A branch and bound algorithm for minimizing the number of crossing arcs in bipartite graphs, Journal of Operational Research 90 (1996) 303–319.
[26] A. Yamaguchi, A. Sugimoto, An approximation algorithm for the two-layered graph drawing problem, in: T. Asano, H. Imai, T. Lee, S. Nakano, T. Tokuyama (Eds.), Proc. International Conference on Computing and Combinatorics, COCOON 1999, in: LNCS, vol. 1627, Springer, 1999, pp. 81–91.