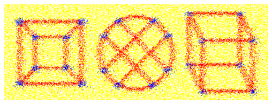

Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 16, no. 2, pp. 151–198 (2012)

Drawing Recurrent Hierarchies

Christian Bachmaier[1] *Franz J. Brandenburg*[1] *Wolfgang Brunner*[1]
Raymund Fülöp[2]

[1]University of Passau, 94030 Passau, Germany
[2]CipSoft GmbH, 93047 Regensburg, Germany

Abstract

Directed graphs are generally drawn as level drawings using the hierarchical approach. Such drawings are constructed by a framework of algorithms which operates in four phases: cycle removal, leveling, crossing reduction, and coordinate assignment.

However, there are situations where cycles should be displayed as such, e. g., distinguished cycles in the biosciences and scheduling processes repeating in a daily or weekly turn. In their seminal paper on hierarchical drawings Sugiyama et al. [31] also introduced recurrent hierarchies. This concept supports the drawing of cycles and their unidirectional display. However, it had not been investigated.

In this paper we complete the cyclic approach and investigate the coordinate assignment phase. The leveling and the crossing reduction for recurrent hierarchies have been studied in the companion papers [3, 4].

We provide an algorithm which runs in linear time and constructs an intermediate drawing with at most two bends per edge and aligned edge segments in an area of quadratic width times the preset number of levels height. This area bound is optimal for such drawings. Our approach needs new techniques for solving cyclic dependencies, such as skewing edges and cutting components. The drawings can be transformed into 2D drawings displaying all cycles counterclockwise around a center and into 3D drawings winding the cycles around a cylinder.

Submitted:	Reviewed:	Revised:	Accepted:
October 2010	March 2011	May 2011	December 2011
	Final:	Published:	
	January 2012	January 2012	
Article type:		Communicated by:	
Regular paper		Michael Kaufmann	

Supported in part by the Deutsche Forschungsgemeinschaft (DFG), grant Br835/15-1.
E-mail addresses: bachmaier@fim.uni-passau.de (Christian Bachmaier) brandenb@informatik.uni-passau.de (Franz J. Brandenburg) brunner@fim.uni-passau.de (Wolfgang Brunner) fueloep@cipsoft.com (Raymund Fülöp)

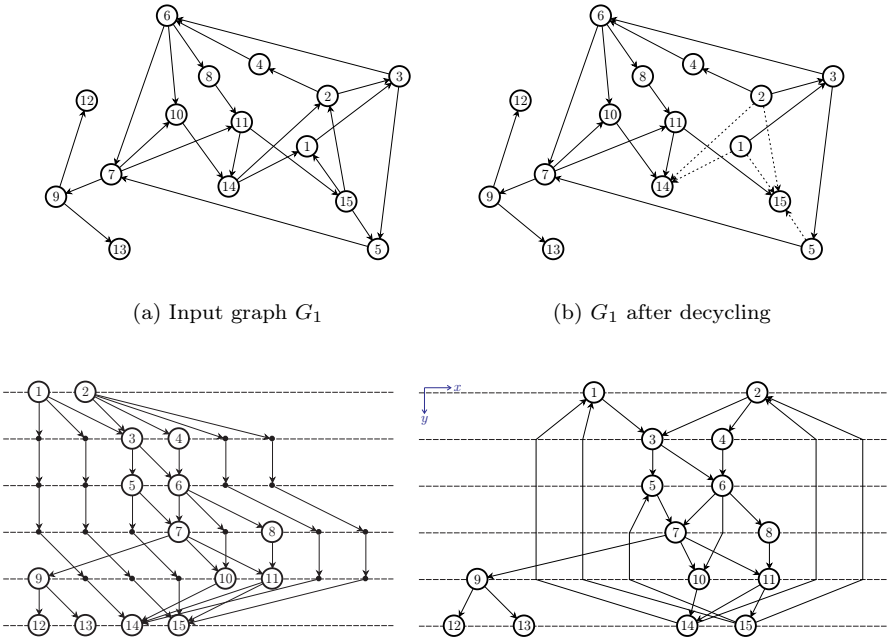
1 Introduction

The hierarchical approach is generally used for poly-line drawings of directed graphs with vertices arranged on horizontal levels. The concept was introduced in 1981 by Sugiyama et al. [31]. It consists of four phases: cycle removal, leveling, crossing reduction, and coordinate assignment. The hierarchical approach has achieved much attention in graph drawing. There are many extensions and improvements and detailed studies of the phases, see [11, 20]. These drawings translate the topological edge direction into a geometric one. It is the appropriate drawing style for directed acyclic graphs, where the drawings have unidirectional edges and reflect the underlying graph as a partial order. Typical applications are schedules, UML diagrams and flow charts, where temporal or causal dependencies are modeled by directed edges and are expressed by a left to right or a downward direction.

In its first phase the hierarchical approach destroys all cycles. However, there are situations where this is unacceptable. For example, there are well-known cycles in the biosciences, where it is a common standard to display them as such. These cycles often serve as a landmark [22]. Cycles are also crucial for repeating processes. Such daily, weekly, or monthly schedules with the same tasks define the periodic event scheduling problem [27]. Again it is important that the cycles are clearly visible. This makes such drawings "nice".

In their seminal paper [31] Sugiyama et al. suggested a solution for the hierarchical style and addressed the cyclic style. The latter is called *recurrent hierarchy*. A recurrent hierarchy is a level graph with additional edges from the last to the first levels. Here, an intermediate drawing is the appropriate drawing style. It is the common hierarchical style with horizontal levels in the plane. However, the first level is duplicated and reappears at the bottom. It is crucial that these two levels are drawn identically, and that there is no disruption or hidden bend for the edges entering the duplicated bottom level. This style supports a continuous vertical scrolling of the drawing. The aesthetic criteria hold for these drawings, in particular, aligned long edges and at most two bends per edge. Typical applications can be found in VLSI design, where it is often necessary to build regular layouts of one-dimensional arrays of identical cells [21].

There are at least two more drawing styles for recurrent hierarchies. Both can be derived from an intermediate drawing, see Fig. 2(b). The first is a 2D drawing, where the levels are *rays* from a common center and are sorted counterclockwise by their number, see Fig. 2(c). All vertices of a level are placed at different positions on their ray, and an edge $e = (u, v)$ is drawn as a monotone counterclockwise poly-spiral curve from u to v wrapping around the center at most once. This style is the best to display cycles as closed curves and to illustrate cyclic dependencies, see Figs. 6(b) and 22. The second style is a 3D drawing on a rolling cylinder, where the levels are horizontal lines and the first level follows the last, see Fig. 2(d). The intermediate and the 3D drawings can be used for an interactive 2D view. A window shows a section of the drawing with some horizontal levels, and it can endlessly be scrolled upwards and downwards,



(a) Input graph G_1

(b) G_1 after decycling

(c) G_1 after leveling and crossing reduction

(d) G_1 after coordinate assignment and postprocessing

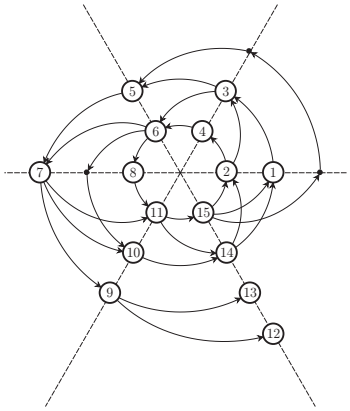
Figure 1: Hierarchical coordinate assignment

see Fig. 2(b). This meets the underlying concept of unidirectional cycles while treating all levels and edges homogeneously.

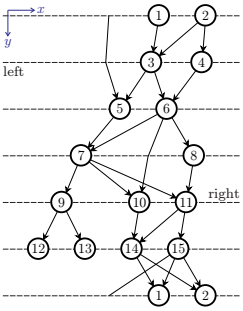
In cyclic drawings all edges are unidirectional, which makes the cycle removal phase vacuous. This saves much effort since the underlying feedback arc set problem is $\mathcal{NP}$-hard [18]. Further advantages over hierarchical drawings are shorter edges and fewer crossings, as Fig. 1(d) illustrates vs. Fig. 2(b). These effects emerge, in particular, if there are back edges from the last to the upper levels, which are long in hierarchical drawings and may cause many crossings.

A planar recurrent hierarchy in 2D style is shown on the cover of the textbook by Kaufmann and Wagner [20]. In their survey paper on the hierarchical approach Bastert and Matuszewski state that "unfortunately this problem (recurrent hierarchies) is still not well studied" [20, page 119]. In fact, recurrent hierarchies had not been studied at all.

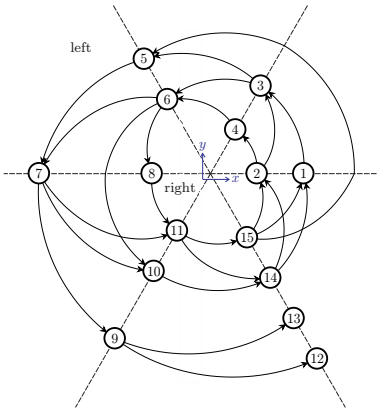
Another 2D drawing of a recurrent hierarchy appears in Sugiyama's textbook [29]. He proposes to draw a recurrent hierarchy by computing a hierarchical layout with emphasis on matching the vertex orders on the uppermost and lowermost levels, using a radial layout of the levels for a clockwise routing of the edges. As our investigations have shown this does not cope with the inherent



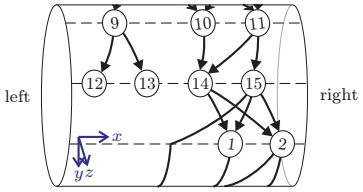
(a) Graph G_1 from Fig. 1(a) after leveling and crossing reduction



(b) Intermediate drawing of (a)



(c) Cyclic 2D drawing of (a)



(d) Cyclic 3D drawing of (a)

Figure 2: Cyclic coordinate assignment (after 4 runs and postprocessing)

problems. For cyclic drawings the algorithmic problems behind the phases are different and mostly algorithmically harder, since the common techniques for leveling, crossing reduction and coordinate assignment fail due to the absence of a well-defined top and bottom of the drawing. The first and last levels are often used for a fresh start and for resolving conflicts, e. g., in the level-by-level sweeps of the third phase.

There are alternative approaches for cyclic drawings of directed graphs. They follow other paradigms which makes them incomparable to our approach. Sugiyama and Misue [30] use a force-based algorithm to obtain a cyclic edge orientation. Pich [23] supplies spectral graph layout techniques to generate cyclic drawings of directed graphs. Both approaches do not draw recurrent hierarchies, since they do not preserve the given leveling of the vertices.

In this work we consider the coordinate assignment phase for cyclic level drawings. Its input is any graph in the proper format, namely an ordered proper cyclic k-level graph. This is the follow-up to our companion papers on the leveling phase [4], a cyclic level planarity test [6], and the crossing reduction [3]. It is part of our frameworks for drawing directed graphs in the hierarchical, radial [1, 2] and cyclic styles, which have been realized in *Gravisto* [5].

There are several algorithms for the coordinate assignment in the hierarchical approach [10, 12, 13, 16, 17, 20, 24–26, 31]. Here we extend the established algorithm of Brandes and Köpf [8] to cyclic level drawings of directed graphs and provide a linear time algorithm using quadratic width and with at most two bends per edge. In the hierarchical approach the algorithm draws long edges vertically aligned. This is not possible any longer since there can be cyclic dependencies in the left-to-right order of the vertices. This particular problem is solved by skewing subgraphs of the drawing and aligning their edges. Cyclic dependencies exclude the exact algorithm of Gansner et al. [16] if in addition to the usual constraints the dummy vertices of a long edge are restricted to the same x-coordinates, such that these are drawn vertically. Then the underlying LP becomes infeasible, as can be seen from the graph G_2 in Fig. 4.

Altogether the cyclic drawing style posed new challenges, which could largely be settled. It gives pleasing drawings for well-suited graphs with cycles.

This paper is organized as follows: After some preliminary definitions in Sect. 2 we first review hierarchical coordinate assignment algorithms in Sect. 3. The main part is the description of our cyclic coordinate assignment algorithm in Sect. 4, which solves the new problem of cyclic dependencies by skewing edges. The time complexity and the area of the drawings are analyzed in Sect. 5. In Sect. 6 cyclic dependencies are settled in a different way. We close with a summary, report on a comparison of the hierarchical and cyclic drawing styles, state some open problems, and show example drawings.

2 Preliminaries

A *cyclic k-level graph* $G = (V, E, \phi)$ where $k \geq 1$ is a directed graph without self-loops with a given surjective level assignment of the vertices $\phi\colon V \to$

$\{1, 2, \ldots, k\}$. Let $V_i \subseteq V$ be the set of vertices v with $\phi(v) = i$. For an edge $e = (u, v) \in E$, u and v are the *start* and *end vertices*. Let $\text{span}(e) = \phi(v) - \phi(u)$ if $\phi(u) < \phi(v)$ and $\text{span}(e) = \phi(v) - \phi(u) + k$, otherwise. An edge e is *short* if $\text{span}(e) = 1$ and *long*, otherwise. A graph is *proper* if all edges are short. Every cyclic level graph can be made proper by adding $\text{span}(e) - 1$ *dummy vertices* for each edge e and thus splitting e into $\text{span}(e)$ many short edges, which are the *segments* of e. In total this may lead to up to $\mathcal{O}(k \cdot |E|)$ new vertices. The first and the last segments of each edge are its *outer segments*, and all other segments between two dummy vertices are its *inner segments*. A proper cyclic k-level graph $G = (V, E, \phi, <)$ is *ordered* if $<$ is a total order for the vertices V_i of each level i with $1 \leq i \leq k$. Let $v_j^{(i)} \in V_i$ denote the j-th vertex on level i and $\text{pos}(v)$ the index of the vertex v on its level, i. e., $\text{pos}(v_j^{(i)}) = j$. In accordance with [8] we say that two segments are in *conflict* in an ordered cyclic level graph if they cross or share a common vertex. Conflicts are of *type 0, 1* or *2* if they are induced by 0, 1, or 2 inner segments, respectively.

We represent drawings of cyclic level graphs by an *intermediate drawing* which is a hierarchical drawing in the plane, where the last level is a one-to-one copy of the first level. Each vertex v is assigned a coordinate $(x(v), y(v))$ with $x(v) \in \mathbb{R}$ and $y(v) = \phi(v) \in \mathbb{N}$. The x-coordinates increase from *left* to *right*, whereas the y-coordinates increase *downwards* in edge direction, see Fig. 2(b). All vertices on level 1 are duplicated on level $k + 1$ using the same x-coordinates. Each segment $s = (u, v)$ is drawn straight-line from $\big(x(u), y(u)\big)$ to $\big(x(v), y(u) + 1\big)$ with *slope* $\frac{1}{x(v) - x(u)}$.

A *2D drawing* as in Fig. 2(c) is obtained from an intermediate drawing by transforming each point $p = (x(p), y(p))$ of the plane to $\big(x_{2\text{D}}(p), y_{2\text{D}}(p)\big) = \big(r(p) \cdot \cos(\alpha(p)), r(p) \cdot \sin(\alpha(p))\big)$ with radius $r(p) = (\text{offset}_x + \max_{v \in V}(x(v))) - x(p) \cdot \delta_x$ and angle $\alpha(p) = (y(p) - 1) \cdot \frac{2\pi}{k}$. The constant offset_x defines the minimum distance of a vertex from the center and δ_x is the minimum distance between vertices on the same level. After the transformation each vertex on level 1 coincides with its copy on level $k + 1$. Note that then right corresponds to the center and left to the outside, see Fig. 2. Accordingly, a *3D drawing* as in Fig. 2(d) uses the coordinates $\big(x_{3\text{D}}(p), y_{3\text{D}}(p), z_{3\text{D}}(p)\big) = \big(x(p) \cdot \delta_x, -r_k \cdot \sin(\alpha(p)), r_k \cdot \cos(\alpha(p))\big)$ for all points p of the drawing, where r_k is the radius of the cylinder. These equations define one-to-one mappings from intermediate to 2D and 3D drawings and transform straight lines to spiral segments.

Finally, a drawing is *(cyclic level) plane* if the edges do not meet or cross except at common endpoints. A (cyclic k-level) graph is *(cyclic level) planar* if such a drawing exists. If there is a plane drawing of an ordered (cyclic) level graph $G = (V, E, \phi, <)$ which respects the order $<$, then we call G *plane*.

3 Foundations

The coordinate assignment is the final phase of the hierarchical approach. The input is an ordered k-level graph. Its vertices are placed on levels where they are

horizontally ordered. The levels determine the y-coordinates. Thus it remains to compute the x-coordinate of each vertex of the input graph and of each dummy vertex for the edge routing. Fig. 1(c) shows the result of the crossing reduction phase for an example graph G_1 and the final drawing is shown in Fig. 1(d).

In the cyclic approach the x-coordinates of the original and the dummy vertices must be computed for the intermediate drawing. This parallels the hierarchical case; however, there are cyclic dependencies, which enforce a skewing. In the 2D case an x-coordinate corresponds to the distance from the center. The ray or the angle is given by the leveling. Fig. 2(a) shows a possible result of the crossing reduction of G_1. Fig. 2(b) is the intermediate drawing with dummy vertices removed. Fig. 2(c) depicts the final 2D drawing, and in Fig. 2(d) the 3D drawing on a cylinder is shown.

The coordinate assignment determines the edge routing. Important drawing conventions and aesthetic criteria are (1) few bends per edge, (2) aligning long edges, (3) a uniform distribution of the vertices and similar distances between a vertex and all its neighbors, and (4) a small width of the drawing. The criteria (1) and (2) are necessary to avoid a "spaghetti" effect [15]. In the hierarchical case long edges are aligned vertically. In cyclic drawings this is not always possible as shall be shown. In the intermediate drawings we align long edges and try to draw them as vertical as possible. They are mapped to concentric circles in 2D drawings.

3.1 Hierarchical Coordinate Assignment

The narrowest drawings are obtained by aligning all vertices leftmost on their levels, as in Fig. 1(c). However, such drawings are not really pleasing, as they have (too) many edge bends and there is no balancing of the vertices among their neighbors. Several better heuristics have been proposed.

Sugiyama et al. [31] suggest a quadratic programming approach for the fourth phase, and propose a priority layout method with several up and down sweeps similar to many crossing reduction techniques. In the down sweeps the x-coordinates of the vertices on the current level are adapted while the x-coordinates on the previous level are fixed. The heuristic has to ensure that the vertex order on the levels remains unchanged. Dummy vertices are prioritized and are moved first for vertically aligned long edges. Heuristics like barycenter [31] and median [14] can be used to position vertices with lower priority.

In their exact approach Gansner et al. [16] model the problem by an LP. The objective function is to minimize the weighted sum of the difference of x-coordinates for adjacent vertices restricted to preserve at least unit distance between vertices on the same level. Inner segments are given the highest weight to draw this lines as close to vertical as possible. Sander [24, 26] uses force directed methods to balance vertices among their neighbors.

Eades et al. [12] spread the vertices of the first and last levels equidistantly on their levels and place all other vertices at the barycenter of their neighbors. They use their algorithm only for special graphs where all sources and sinks

are on the extreme levels. Note that this approach may change the computed vertex orders on the levels.

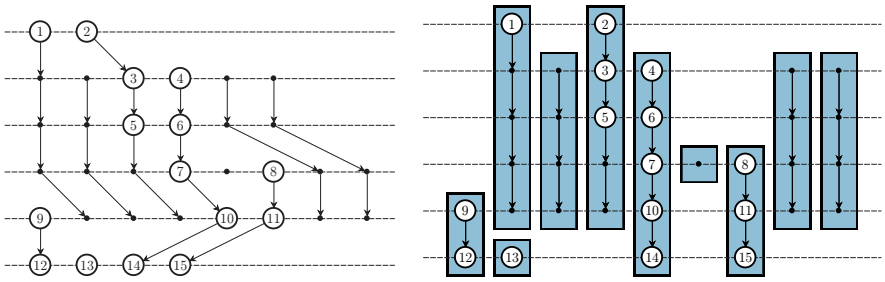
The algorithm of Buchheim et al. [10] consists of two phases. First the x-coordinates of dummy vertices are computed and then considering these as fixed the non-dummy vertices are placed between them. It guarantees at most two bends per edge and draws inner segments vertically. The running time is $\mathcal{O}((|V| + |E|) \cdot (\log(|V| + |E|))^2)$ for a leveled graph $G = (V, E)$ containing dummy vertices and only short edges.

3.1.1 Algorithm by Brandes and Köpf

Our favorite algorithm for the hierarchical coordinate assignment is the one by Brandes and Köpf [8], since it runs in linear time and the resulting drawings are of high quality. They have a good resolution, i. e., integral coordinates, if the horizontal separation between vertices is even. Most importantly, they guarantee at most two bends per edge, which in addition occur at the first and last levels properly crossed by an edge. Moreover, the inner segments of long edges are vertically aligned and the vertices are balanced over their neighbors. These aesthetics shall be adopted. We describe it in more detail here, as our cyclic coordinate assignment algorithm is an extension to cyclic level graphs.

The algorithm uses a thinning strategy. Whenever possible vertices are aligned with their median neighbors extracting all other incident segments. The alignment is directed towards the median of its incoming (align upwards) or outgoing neighbors (align downwards). If the number of neighbors is even, there is a left (align left) and a right median (align right). Hence, the alignment and the subsequent phase is executed four times, once for each combination of upwards and downwards with left and right alignment. We call each such execution one *run* and describe the upward alignment to the left only. The others are obtained by flipping the ordered k-level graph horizontally and/or vertically.

Segments of the graph are extracted until each vertex has at most one incoming and one outgoing segment and there are no crossings in a drawing respecting the order of the vertices on each level. The crossing of an outer and an inner segment is a type 1 conflict. In that case the outer segment is removed. By assumption type 2 conflicts are excluded, which are crossings of two inner segments. Thus all inner segments are kept. The remaining type 0 conflicts correspond to a pair of outer segments which either cross or share a common vertex. They are resolved greedily in a leftmost fashion. First, it is tried to align a vertex with its left median incoming neighbor, otherwise, with the right median. An alignment fails if the used segment causes a crossing with an already aligned segment or its end vertex was already used for another alignment. The remaining segments build paths, which are called *blocks* and are drawn vertically in the hierarchical case. For a horizontal compaction the blocks are sorted topologically according to the left-to-right order on the levels. So the x-coordinate of each (dummy) vertex is obtained in the left upwards run. In [8] the results of the four runs are finally balanced out by taking the average of the two median x-coordinates for each vertex.



(a) Remove edges to align vertices (b) Compact blocks

Figure 3: The algorithm of Brandes and Köpf [8] for G_1 from Fig. 1(c)

As an example consider the graph in Fig. 1(c). The graph in Fig. 3(a) is obtained by aligning vertices upwards to the left and removing the other segments. Each path in the graph is a block, which is compacted in Fig. 3(b). The final drawing results from the balancing step of the four drawings and is shown in Fig. 1(d).

3.1.2 Postprocessing

After the computation of all x-coordinates some additional steps are taken to clean up the drawing. All dummy vertices are removed and if necessary are replaced by edge bends. Edges are redirected to their original direction if they were reversed in the decycling phase. Lastly, the computed x- and y-coordinates are multiplied with user-given values to scale the drawing to the desired size.

4 Cyclic Coordinate Assignment

In this section we describe the coordinate assignment phase for cyclic level graphs. We extend the algorithm of Brandes and Köpf [8] and use their notation. The input for our algorithm is an ordered proper cyclic level graph. It can be obtained by running the earlier phases of the framework. As in [8] we assume that there are no type 2 conflicts after the crossing reduction phase. This is guaranteed, e. g., by the global sifting algorithm for crossing reduction [3]. The algorithm has no further preconditions and does not change the levels and the orderings of the vertices on the levels. Note that dummy vertices have been introduced after the leveling.

Algorithm 1 consists of three basic steps: block building (lines 4–5), horizontal compaction (lines 6–12), and balancing (line 14), which correspond to the according steps in [8]. The first two steps are carried out four times with runs for each combination of a left/right and an upwards/downwards alignment (line 2). The four results are merged by the balancing step. We only describe

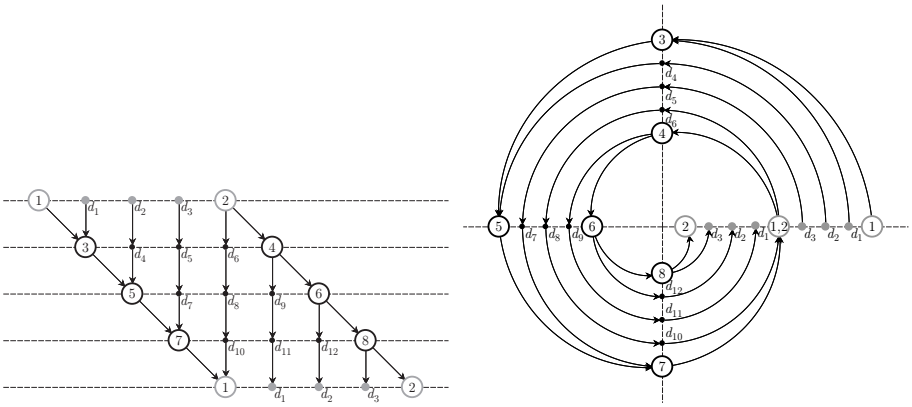
Algorithm 1: cyclicCoordinateAssignment

Input: An ordered proper cyclic k-level graph $G' = (V', E', \phi', <)$
Output: Coordinates $(x(v), y(v))$ for each $v \in V'$ in the intermediate drawing $\mathcal{I}$

```
1  P ← ∅
2  foreach (h, v) ∈ {left, right} × {upwards, downwards} do
3      G'' ← flip(G', h, v)                    // according to the current run
4      H ← buildCyclicBlockGraph(G'')
5      H ← normalize(H)                        // split long and closed blocks
6      S ← computeSCCs(H)
7      foreach complex SCC S ∈ S do
8          (S', E_I) ← cutSCC(S)               // returns the hierarchical block graph
9          width(S') ← compactBlocks(S', E_I)
10         skew(S', −(wind(S') · k)/ width(S'))   // skew S' with given slope
11         S ← S \ S ∪ S'
12     compact(S)                              // globally all SCCs
13     P ← P ∪ flip(S, h, v)
14 I ← balance(P)                              // balance four runs
15 return I
```

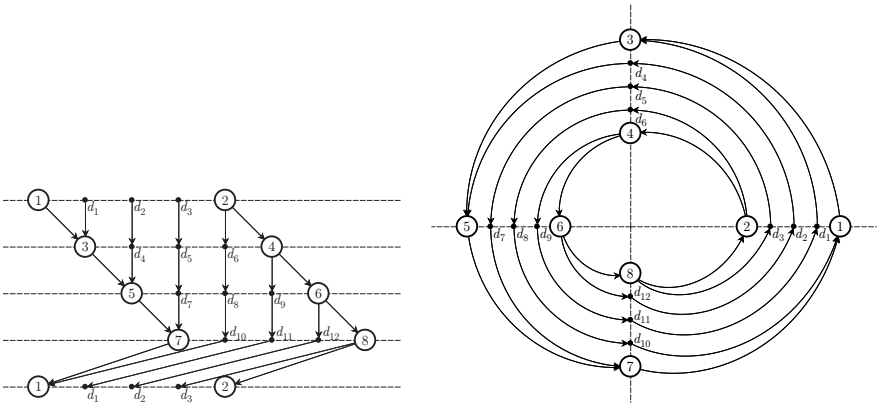
the run for the upwards alignment to the left. The other three runs are realized by flipping the graph horizontally and/or vertically before and after each run (lines 3, 13). The thereby computed intermediate drawing can be transformed into the 2D or 3D drawings. Bends occur at dummy vertices which are removed thereafter.

In the cyclic case there may be unavoidable cyclic dependencies in the left-to-right order among vertically aligned paths. This is the major new challenge in the cyclic coordinate assignment. For an illustration consider the graph G_2 in Fig. 4 to Fig. 6. In each figure an intermediate drawing of the same graph is shown on the left and the corresponding cyclic drawing is shown on the right. If all inner segments were drawn vertically (see Fig. 4), this would lead to the cyclic dependency $x(d_1) < x(d_2) < x(d_3) = x(d_5) = x(d_7) < x(d_8) < x(d_9) = x(d_{11}) = x(d_1)$, which is a contradiction. We will later formalize these cyclic dependencies and call them rings. In the intermediate drawing (see Fig. 4(a)), the vertices of the two copies of the first level do not have the same x-coordinate. The two copies of these vertices cannot be drawn at the same position in the cyclic drawing (see Fig. 4(b)). This problem could be solved by additional bends, e. g., between the last and the first levels. Then the two copies of each vertex on level 1 have the same x-coordinate (see Fig. 5(a)) and there is a cyclic drawing (see Fig. 5(b)). However, then edges can have up to four bends, and the edges from the last to the first level need a special treatment. This is a drawing one would get from a hierarchical layout with a simple matching of the vertices on the levels 1 and $k + 1$ as proposed in [29]. Another possibility is to



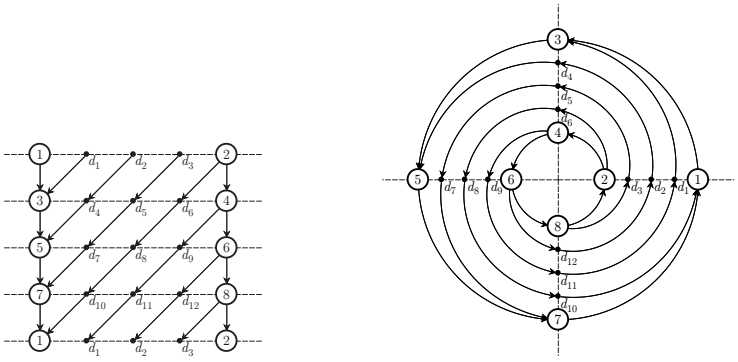
(a) Intermediate drawing (b) Cyclic drawing

Figure 4: Graph G_2 with vertical inner segments



(a) Intermediate drawing (b) Cyclic drawing

Figure 5: Graph G_2 with additional bends



(a) Intermediate drawing (b) Cyclic drawing

Figure 6: Graph G_2 with skewed inner segments

align inner vertices using the same slope (see Fig. 6). Then there are at most two bends per edge, and the symmetry of the graph is reflected by the drawing. We pursue this approach and solve the associated problems, i. e., computing the slopes and determining how many different slopes are needed.

There are means to avoid cyclic dependencies. For example the global sifting heuristic for the crossing reduction from [3] can be used together with a slight modification of our coordinate assignment algorithm. However, this is at the expense of more crossings, less balanced drawings, and the loss of generality. We will provide a particular solution in Sect. 6. In order to draw any given ordered cyclic level graph, we need an algorithm which can handle cyclic dependencies. As an example consider the graph in Fig. 6, which can only be drawn without crossings with the permutations of each level as given in Fig. 6 or their reversal. These permutations are the result of an optimal crossing reduction, e. g., a cyclic adaption of [19] or a planarity testing and embedding algorithm [6]. To avoid cyclic dependencies other permutations on the levels are needed which result inevitably in crossings.

4.1 Block Building

The algorithm of Brandes and Köpf [8] builds blocks by the selection of paths. We follow this approach with extensions for the cyclic structure. We try to align each vertex with the median of its upper neighbors by assigning them to the same block. Removing all other non-aligned segments results in a graph, where each vertex has at most one incoming and one outgoing segment and thus consists of paths and cycles. Such graphs are called cyclic path graphs.

Definition 1 *A* cyclic path graph $H' = (V, E_{\text{intra}}, \phi, <)$ *is a plane ordered proper cyclic k-level graph. Each vertex of* H' *has in-degree and out-degree*

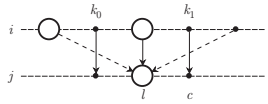


Figure 7: Marking type 1 conflicts

at most one. We call each connected component of H' a block *and all edges $e \in E_{\text{intra}}$* intra-block edges*. A block B is* closed *if each vertex of B has in-degree and out-degree one and B is* open*, otherwise. The* height *of B is defined as the number of intra-block edges in B. Let* $\text{block}(v)$ *be the block of a vertex $v \in V$, and* $\text{levels}(B)$ *the set of levels on which B has (dummy) vertices. For an open block B let* $\text{top}(B)$ *and* $\text{bottom}(B)$ *be the topmost and lowermost vertex, i. e., the vertices with in-degree and out-degree zero, respectively. Let $\mathcal{B}$ denote the set of all blocks.*

The cyclic block graph $H = (V, E_{\text{intra}} \dot\cup E_{\text{inter}}, \phi)$ *of H' is obtained by adding an* inter-block edge $e \in E_{\text{inter}}$ *from each vertex in H' to its consecutive right vertex on the same level. For a vertex $v \in V$ let* $\text{left}(v)$, $\text{right}(v)$, $\text{up}(v)$, *and* $\text{down}(v)$ *be the start vertex of the incoming inter-block edge, the end vertex of the outgoing inter-block edge, the start vertex of the incoming intra-block edge, and the end vertex of the outgoing intra-block edge, respectively. A cyclic block graph is* normalized *if it consists of open blocks of height at most $k - 1$.*

A hierarchical path graph *is defined analogously being an ordered proper (non-cyclic) level graph. A* hierarchical block graph *is a hierarchical path graph extended by inter-block edges.*

The drawing of the cyclic block graph determines the coordinates of all (dummy) vertices in the final drawing of the original graph. As all segments of the original graph are straight lines, the drawing is completely determined. Algorithm 2 constructs a cyclic block graph from an ordered proper cyclic k-level graph. Each level is traversed separately (line 3). First, outer segments involved in type 1 conflicts between the current level j and its predecessor level i are marked. For that traverse the lower level from left to right (lines 7–18) and stop at each end vertex of an inner segment and after the last vertex on the level. Traverse all (non-dummy) vertices l between the last and the current inner segment on level j (lines 13–17). If an incoming segment of the non-dummy vertex l starts left of the last found inner segment ($k < k_0$) or starts right of the current inner segment ($k > k_1$), it is marked as it crosses at least one inner segment. See Fig. 7 for an illustration of this situation where marked segments are dashed.

Next traverse the current level j from left to right again and try to align each vertex $v_c^{(j)}$ with one of its median predecessors (lines 20–28). First, try its upper left median, then its upper right median (lines 24–28). The second test is skipped if the first was successful (line 25). A segment cannot be used for aligning if it is marked or if it would cross a segment already used for aligning. In the latter case both segments are outer segments and are in a type 0 conflict.

Algorithm 2: buildCyclicBlockGraph

Input: An ordered and proper cyclic k-level graph $G' = (V', E', \phi', <)$
Output: The cyclic block graph H of G' (left upper run)

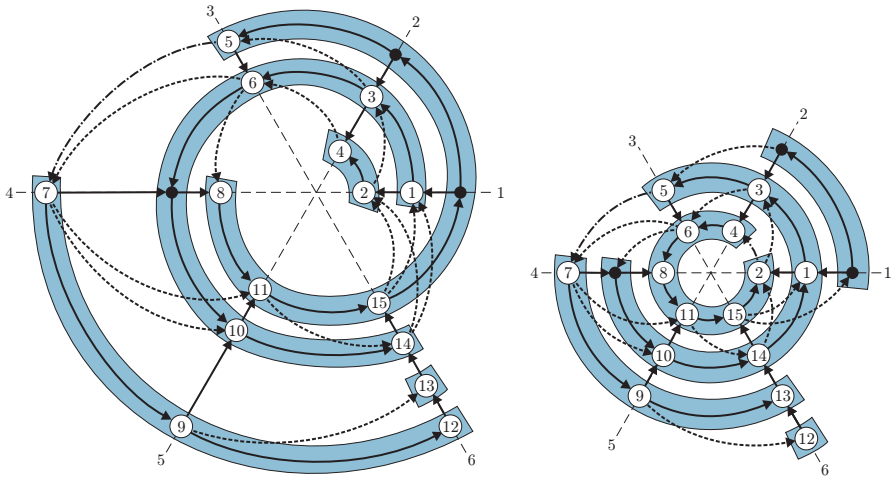
```
1  E_intra ← ∅
2  E_inter ← ∅
3  for 1 ≤ i ≤ k do
4      j ← (i mod k) + 1                    // next level after i
5      k_0 ← 0
6      l ← 1
7      for 1 ≤ c ≤ |V'_j| do
8          if c = |V'_j| or v_c^(j) is an end vertex of an inner segment then
9              if v_c^(j) is an end vertex of an inner segment s then
10                 k_1 ← position of the start vertex of s in V'_i
11             else
12                 k_1 ← |V'_i|
13             while l ≤ c do
14                 foreach upper neighbor v_k^(i) of v_l^(j) do
15                     if k < k_0 or k > k_1 then
16                         mark the segment (v_k^(i), v_l^(j))
17                 l ← l + 1
18             k_0 ← k_1
19     r ← 0
20     for 1 ≤ c ≤ |V'_j| do
21         if c < |V'_j| then
22             E_inter ← E_inter ∪ {(v_c^(j), v_{c+1}^(j))}
23         if v_c^(j) has upper neighbors u_1 < ... < u_d with d > 0 then
24             for ⌊(d+1)/2⌋ ≤ m ≤ ⌈(d+1)/2⌉ do
25                 if up(v_c^(j)) = null then          // v_c^(j) is not yet aligned
26                     if (u_m, v_c^(j)) is not marked and r < pos(u_m) then
27                         E_intra ← E_intra ∪ {(u_m, v_c^(j))}     // up(v_c^(j)) ← u_m
28                         r ← pos(u_m)
29 H ← (V', E_intra ∪̇ E_inter, φ')
30 return H
```

Algorithm 3: normalize

Input: A cyclic k-level block graph $H = (V, E_{\text{intra}} \dot\cup E_{\text{inter}}, \phi)$ with blocks $\mathcal{B}$

Output: A cyclic block graph H with all blocks of height at most $k-1$

```
1  for B ∈ 𝓑 do
2      if B is a closed block then
3          remove an arbitrary outer segment of B from E_intra
4      else if height(B) ≥ k then
5          v ← top(B)
6          v_0 ← ∅
7          w ← 0
8          w_0 ← 0
9          while down(v) ≠ null do
10             w ← w + 1
11             if (v, down(v)) is an inner segment then
12                 v_0 ← v
13                 w_0 ← w
14             if w = k then
15                 E_intra ← E_intra \ {(v_0, down(v_0))}
16                 w ← w − w_0
17             v ← down(v)
18 return H
```



(a) Left upper run (b) Right upper run

Figure 8: Block graphs of G_1 from Fig. 2(a) as cyclic 2D drawings after normalization; dashed edges are absent in the block graph. We prefer 2D drawings to display the cyclic dependencies

The current vertex $v_c^{(j)}$ becomes the top vertex of a new block if both alignments fail or if the vertex does not have any upper neighbors. Outer segments involved in type 1 conflicts are never used for aligning and there are no type 2 conflicts. Hence, all inner segments of the graph are kept for aligning. The invariants that there are at most two bends per edge and that inner segments are aligned are guaranteed by the fact that all inner segments of an edge are in one block which is drawn with a fixed slope. All segments not used for aligning are removed to obtain the cyclic path graph. E_{intra} is the set of all remaining segments. See Fig. 8 for examples of cyclic block graphs of G_1. Vertices and intra-block edges of the same block are framed. The dotted segments have been removed in the block building phase by Algorithms 2 and 3. The cyclic block graph contains the additional inter-block edges which lie on the level lines pointing to the center. Note that there are no crossings in a drawing of the cyclic block graph which respects the order of the vertices on each level.

The cyclic block graph can have closed blocks (with height k) and open blocks with height $\geq k$ (*spirals*). To simplify the coordinate assignment we normalize blocks in Algorithm 3 to a height of at most $k-1$. In a closed block removing an arbitrary outer segment yields an open block of height $k-1$ (lines 2–3). In a spiral we traverse the block from the topmost to the lowest segment and iteratively remove the latest possible outer segment such that the block above has height at most $k-1$ (lines 4–17). In both cases, such outer segments always exist as no edge can span more than k levels. So the invariant of at most

two bends per edge is preserved. Note that an originally closed block is not skewed as are other blocks in Sect. 4.2. It cannot be part of a cyclic dependency since it splits the graph into two disjoint parts. An open block of height at least k would have to be skewed anyway as it contains a cyclic dependency. See Fig. 8 as an example: In both runs the edge $(5, 7)$ has been removed to split a long block into two shorter ones. In the right upper run the segment $(2, 4)$ has been removed to open a closed block. The result is a cyclic block graph with open blocks of height at most 5.

4.2 Horizontal Compaction

In this section we compact the cyclic block graph by arranging all blocks as close to each other as possible to reduce the width of its drawing. Not all blocks can be drawn vertically, as there may be cyclic dependencies in the left-to-right order among blocks, which we call rings. A ring is a cycle in the block graph where the direction of the inter-block edges is preserved and the intra-block edges are used in any direction.

Definition 2 *Let $H = (V, E_{\text{intra}} \dot\cup E_{\text{inter}}, \phi)$ be a cyclic block graph. A* block path *P in H is a sequence of vertices $v_1, \ldots, v_s \in V$ such that for each pair of consecutive vertices v_i and $v_{i+1}, 1 \le i < s$, $(v_i, v_{i+1}) \in E_{\text{intra}}$ or $(v_{i+1}, v_i) \in E_{\text{intra}}$ or $(v_i, v_{i+1}) \in E_{\text{inter}}$. It is* simple *if all vertices are distinct. A block path is a* ring *R if $v_1 = v_s$ and it traverses at least one inter-block edge.*[1] *A ring is* simple *if the vertices $v_1, \ldots, v_{s-1}$ are distinct. The* width *of R is the number of its traversed inter-block edges. Let c_{down} and c_{up} be the number of intra-block edges traversed in R in and against their direction, respectively. The* number of windings *of R is defined by* $\text{wind}(R) = \lfloor (c_{down} - c_{up})/k \rfloor \in \mathbb{Z}$.

We traverse intra-block edges in both directions and regard each block as strongly connected. Using both directions is essential to finding a ring with maximum width. Reconsider the drawing in Fig. 4(a) and the cyclic dependency of the vertices' x-coordinates $x(d_1) < x(d_2) < x(d_3) = x(d_5) = x(d_7) < x(d_8) < x(d_9) = x(d_{11}) = x(d_1)$. In the cyclic block graph of this graph the sequence of vertices $d_1, d_2, d_3, d_5, d_7, d_8, d_9, d_{11}, d_1$ is a ring. Each inequality represents an inter-block edge traversed from left to right and each equality represents an intra-block edge traversed upwards or downwards. The ring has width 4 as four inter-block edges are used. This corresponds to the fact that the lower copy of level 1 is shifted four units to the right versus level 1 in the drawing in Fig. 4(a). A ring needs at least one inter-block edge, otherwise also the sequence d_3, d_5, d_7, d_5, d_3 would be a ring. However, such a sequence does not raise a problem, as the block is never left and the same x-coordinate can be assigned to all vertices of the sequence. Since we split spirals the start and end vertices of each inter-block edge lie in different blocks and, hence, each ring traverses at least two blocks.

[1] The latter ensures that R "wraps round the center" and is not only a single vertex/block.

Informally, wind(R) counts how often a ring R wraps around the center or the cylinder in a drawing. As each ring is an ordered sequence we count windings along increasing and decreasing levels positively and negatively, respectively. The reachability of vertices using block paths partitions the block graph into strongly connected components (SCCs) which we consider separately. There are two types of SCCs: The *simple SCCs* consist of a single block. All other SCCs are *complex* and contain rings. Figure 8(a) consists of three simple SCCs ($(2,4)$, $(7,9,12)$ and (13)) and one complex SCC (the remaining two blocks), whose simple rings R have width(R) $= 2$ and wind(R) $= 1$. For rings some properties can be established:

Lemma 1 *A hierarchical block graph G does not contain a ring.*

Proof: The proof is by constructing real x-coordinates for each vertex such that all vertices in the same block have the same x-coordinate. Assign x-coordinate j to the j-th vertex on level 1. For each level $i > 1$ and each vertex v on level i, which is in the same block as a vertex u on level $i-1$, assign v the x-coordinate $x(v) = x(u)$. This does not contradict the order on level i as there are no type 2 conflicts. Let $v_1, \ldots, v_l$ be the vertices on level i which have been assigned x-coordinates so far. For each $1 \leq m < l$ assign all vertices between v_m and v_{m+1} increasing x-coordinates between $x(v_m)$ and $x(v_{m+1})$ respecting the order on level i. Finally, assign all vertices left of v_1 and all right of v_l x-coordinates respecting the order smaller than $x(v_1)$ and larger than $x(v_l)$, respectively.

This procedure computes x-coordinates for all vertices, which coincide for all vertices of each block. Suppose that G contains a ring $R = (v_1, \ldots, v_{s-1}, v_1)$. Then $x(v_a) = x(v_b)$ for every intra-block edge (v_a, v_b) in R and $x(v_c) < x(v_d)$ for every inter-block edge (v_c, v_d) in R. According to Definition 2 R contains at least one inter-block edge (v_f, v_g). Thus $x(v_1) \leq x(v_f) < x(v_g) \leq x(v_s) = x(v_1)$, which is a contradiction. □

Lemma 2 *If R is a ring of a cyclic block graph of an ordered proper cyclic level graph G, then* wind(R) $\neq 0$.

Proof: Assume that R is a ring in the cyclic block graph of G with wind(R) $= 0$. Then R contains the same number c of intra-block edges traversed in and against their direction. Thus in any cyclic 2D drawing respecting the vertex order R may wrap at most $\frac{c}{k}$ times around the center but returns in the opposite direction without enclosing it. Unwrap G a number of $l = \lceil \frac{c}{k} \rceil + 1$ times by placing multiple copies of the intermediate drawing one below the other and merging first and last levels. This gives an ordered hierarchical level graph $G_l = (V_l, E_l, \phi_l, <_l)$ such that its block graph H_l completely contains R as a connected subgraph R_l. If R_l is a circle it is a ring contradicting Lemma 1. If R_l is a block path, then R_l starts at a vertex u and ends at a copy u' of u with $\phi_l(u) \neq \phi_l(u')$. Thus R_l does not contain the same number of intra-block edges traversed in and against their direction, which is a contradiction to wind(R) $= 0$. □

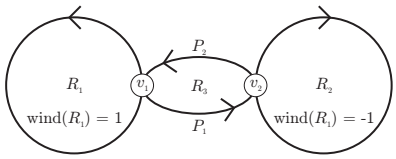


Figure 9: Schematic construction of the proof of Theorem 1 with $\text{wind}(S) > 0$

Lemma 3 *If R is a simple ring of a cyclic block graph, then* $|\text{wind}(R)| \leq 1$.

Proof: Suppose there is a simple ring R with $|\text{wind}(R)| > 1$. As R wraps around the center in any cyclic 2D drawing respecting the vertex order more than once, the corresponding curve crosses itself in any drawing of R. As R is simple the crossing cannot be at a common vertex.

Each cyclic path graph is plane. Further, each drawing of it respecting its order can be extended to a plane drawing of its cyclic block graph by adding the inter-block edges along the level lines. Since R is a subgraph of the cyclic block graph, this is a contradiction. □

Theorem 1 *Let $\mathcal{R}$ be the set of all simple rings of an SCC in a cyclic block graph. Then all rings $R \in \mathcal{R}$ have the same winding number* $\text{wind}(R)$*, either* 1 *or* -1.

Proof: According to Lemmas 2 and 3, $|\text{wind}(R)| = 1$ holds for each ring $R \in \mathcal{R}$. Suppose that there are two simple rings $R_1, R_2 \in \mathcal{R}$ with $\text{wind}(R_1) = 1$ and $\text{wind}(R_2) = -1$, see Fig. 9. Let $v_1 \in R_1$ and $v_2 \in R_2$ be vertices in different blocks. Such vertices exist as each ring traverses at least two blocks. Let P_1 be a block path from v_1 to v_2 and P_2 be a block path from v_2 to v_1. Both exist due to the strong connectivity. Concatenating P_1 and P_2 results in a (not necessarily simple) ring R_3 through v_1 and v_2. By Lemma 2 $\text{wind}(R_3) \neq 0$. If $\text{wind}(R_3) > 0$, let R_4 be a non-simple ring consisting of R_3 and $\text{wind}(R_3)$ many copies of R_2 joined at v_2. Otherwise, let R_4 be a ring consisting of R_3 and $-\text{wind}(R_3)$ many copies of R_1 joined at v_1. In both cases $\text{wind}(R_4) = 0$, which contradicts Lemma 2.

□

Hence, all simple rings of an SCC S have the same winding number which is taken as the winding number of S.

Definition 3 *For a complex SCC S of a cyclic block graph containing a simple ring R define* $\text{wind}(S) = \text{wind}(R)$ *and let* $\text{width}(S)$ *be the maximum width of all simple rings in S.*

4.2.1 Compaction of a Complex Strongly Connected Component

In the intermediate drawing it is impossible to draw all blocks of a complex SCC S straight-line and vertically. However, it is possible to draw all blocks of

S with the same slope. The slope has to be chosen such that the curve of each ring in S starts and ends at the same coordinate. All simple rings of S have the same number of windings $\text{wind}(S)$, which is either 1 or -1. Traversing a simple ring R visits all k levels and uses $\text{width}(R)$ inter-block edges. To draw R we could use the slope $-(\text{wind}(R) \cdot k)/\text{width}(R)$, which would result in inter-block edges of unit length. In order to draw all blocks of S with the same slope (line 10 in Algorithm 1), we must use the width of the widest simple ring of S and the slope $-(\text{wind}(S) \cdot k)/\text{width}(S)$. With this slope the widest ring fits exactly and uses unit length inter-block edges. All narrower rings have some unused horizontal space in the drawing and have inter-block edges which are longer than one unit. As an example consider Fig. 6 again: The inner segments form an SCC S with $\text{width}(S) = 4$ and $\text{wind}(S) = 1$ and the graph uses $k = 4$ levels. Hence, the slope of $-(\text{wind}(S) \cdot k)/\text{width}(S) = -1$ is used for these inner segments in Fig. 6(a). Note that the positive y-axis points downwards in intermediate drawings. These properties are comprised by slope alignment.

Definition 4 *Let S be an SCC of a cyclic block graph. An intermediate or 3D drawing of S is* slope aligned *if the blocks are drawn as straight lines with the same slope, vertices have at least unit distance in x-direction, and the left-to-right order of the vertices is preserved. A 2D drawing of S is* slope aligned *if the corresponding conditions hold. Then blocks are drawn as spirals, where the change in the angle over the change in the radius of all blocks is a constant. We call this constant* slope *as well.*

A drawing of a block graph is slope aligned *if its SCCs are drawn slope aligned and the vertex order and unit distance is preserved. It is* uniformly slope aligned *if additionally only three slopes $-s, \infty, s$ are used for all SCCs.*

Let Γ be a drawing of an ordered cyclic k-level graph which is obtained from the drawing of a single block graph H by inserting the deleted edge segments. Then Γ is (uniformly) slope aligned *if so is the drawing of H.*

For the slope of the SCC S we have to compute $\text{wind}(S)$ and $\text{width}(S)$. $\text{wind}(S)$ is determined by $\text{wind}(R)$ for an arbitrary ring R in S. Hence, it suffices to find a simple ring R in S, which can easily be done in linear time. In the sequel we shall assume $\text{wind}(S) = 1$. Computing $\text{width}(S)$ corresponds to determining the width of the widest simple ring in S. In general, the problem of finding the longest cycle in a directed graph is $\mathcal{NP}$-hard [18]. But complex SCCs are very special graphs, which can be drawn without crossings with the given leveling. Finding the widest simple ring can be solved in linear time by cutting the SCC as the following section shows.

4.2.2 Cutting a Strongly Connected Component

The idea is to cut an SCC S from a leftmost vertex to a rightmost vertex on their levels along a special block path in the cyclic block graph. Computing the width of S and compacting the layout is done simultaneously.

Lemma 4 *Let S be an SCC of a cyclic block graph with* $\text{wind}(S) = 1$. *Then S contains a block B where the lowest vertex $v = \text{bottom}(B)$ is the leftmost vertex of the vertices of S on its level $\phi(v)$.*

Proof: Suppose that the lowest vertex of each block of S has a left neighbor in S. Then it has an incoming intra-block edge. Starting at any vertex traverse each block downwards to its lowest vertex and then against the direction of inter-block edges to a new block. This can be repeated and ends in a circle through some vertex v since there is no sink. The reversal of this circle is a simple ring R with $\text{wind}(R) = -1$, since all intra-block edges are used against their direction. However, $\text{wind}(S) = 1$, which contradicts Theorem 1. □

Recall that we consider the case $\text{wind}(S) = 1$. To cut the SCC S Algorithm 4 finds a path P starting at a leftmost vertex v on its level which is the lowest vertex in its block B (line 1). Such a vertex and block always exist by Lemma 4. It traverses the block to the topmost vertex and uses the outgoing inter-block edge of that vertex to enter a new block. This procedure is repeated until a block is reached, whose topmost vertex does not have an outgoing inter-block edge and therefore is a rightmost vertex on its level (lines 4–12). This process will terminate, as otherwise a vertex u would have been visited twice. The sequence between the two occurrences of u is a ring R with $\text{wind}(R) = -1$, which contradicts $\text{wind}(S) = 1$. Hence, P is a path from a leftmost vertex to a rightmost vertex.

Let P consist of the vertices $v_1, \ldots, v_m$. For each vertex v_i $(i > 1)$ we remove its incoming inter-block edge if it does not start at v_{i-1} (lines 9–10). In other words, we remove all incoming inter-block edges not belonging to P and thus all those edges left of P. Note that P is only constructed for clarity in Algorithm 4 and not needed otherwise, as we cut the inter-block edges while traversing from left to right.

We construct a hierarchical block graph S' corresponding to the cyclic block graph in the following way: Initially we assign the level $\phi'(v) = \phi(v)$ (line 15) to the chosen leftmost vertex v. In a traversal of the block graph we assign each vertex a new level ϕ': Using an inter-block edge (in any direction) we assign both end vertices the same level ϕ' (lines 19–24). Using an intra-block edge in or against its direction we increase (lines 28–30) or decrease (lines 25–27) the level ϕ' by 1, respectively, without using a modulo operation.

See Fig. 10(a) for an example. Cutting the SCC starts at the bottom of the black block and the dashed path crosses all removed inter-block edges. Fig. 10(b) shows the resulting hierarchical block graph, where we allow negative levels for simplicity.

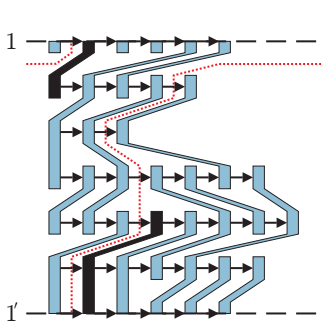
The removed incoming inter-block edges left of P cut all simple rings in S exactly once as the following lemma shows.

Lemma 5 *Let S be a complex SCC of a cyclic block graph with* $\text{wind}(S) = 1$. *If the inter-block edges E_I computed by Algorithm 4 are removed from S, then exactly one inter-block edge is taken from each simple ring in S.*

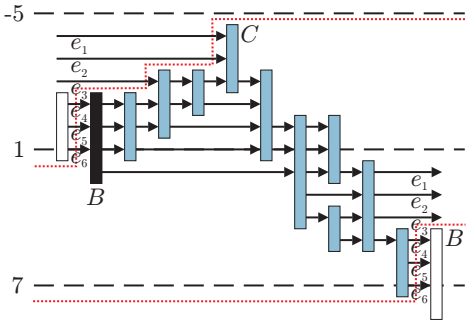
Algorithm 4: cutSCC

Input: An SCC $S = (V, E_{\text{intra}} \dot\cup E_{\text{inter}}, \phi)$ of a cyclic k-level block graph
Output: A hierarchical block graph S' and the set E_I of removed inter-block edges

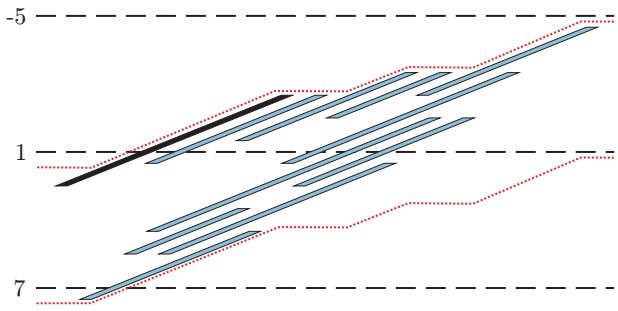
```
1  u ← v ← any leftmost vertex on its level with v = bottom(block(v))
2  P ← ∅
3  E_I ← ∅
4  while u ≠ null do
5      P ← P ∘ (u)
6      if up(u) ≠ null then
7          u ← up(u)
8          if left(u) ≠ null then
9              E_inter ← E_inter \ {(left(u), u)}
10             E_I ← E_I ∪ {(left(u), u)}
11     else
12         u ← right(u)
13 for u ∈ V do                    // initialize leveling of hierarchical block graph
14     φ'(u) ← null
15 φ'(v) ← φ(v)
16 Q ← {v}
17 while Q ≠ ∅ do
18     u ← Q.remove()
19     if left(u) ≠ null and φ'(left(u)) = null then
20         φ'(left(u)) ← φ'(u)
21         Q.add(left(u))
22     if right(u) ≠ null and φ'(right(u)) = null then
23         φ'(right(u)) ← φ'(u)
24         Q.add(right(u))
25     if up(u) ≠ null and φ'(up(u)) = null then
26         φ'(up(u)) ← φ'(u) − 1
27         Q.add(up(u))
28     if down(u) ≠ null and φ'(down(u)) = null then
29         φ'(down(u)) ← φ'(u) + 1
30         Q.add(down(u))
31 S' ← (V, E_intra ∪̇ E_inter, φ')
32 return (S', E_I)
```



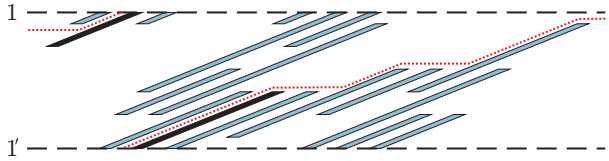
(a) Left-aligned intermediate drawing



(b) Compacted drawing



(c) Skewed drawing



(d) Final intermediate drawing

Figure 10: Drawing of a complex SCC with the dotted line as cut

Proof: Consider any cyclic 2D drawing Γ of S respecting the order of the vertices, e. g., a leftmost drawing. Algorithm 4 computes a block path P which connects a leftmost vertex v_1 and a rightmost vertex v_m. Every vertex $v_i \in P$ has three possibilities for an incident edge e in S with respect to the traversing direction of P from v_1 to v_m: e can be left of P, right of P or part of P. As Γ is plane no edge crosses P. Let R be a simple ring in S. Since $wind(R) = 1$ it has at least one common vertex v_j with P. Deleting the incident edges of v_j left of P removes exactly one edge in R. As P only traverses inter-block edges in their direction and intra-block edges against their direction, edges incident to v_j and left of P can only be incoming edges $(u, v_j) \in E_{\text{inter}}$ or $(w, v_j) \in E_{\text{intra}}$. The latter is not possible by the construction of P, because otherwise the next edge $e = (v_j, \cdot)$ in P would be an inter-block edge where v_j is not the topmost vertex of its block as $\text{up}(v_j) = w$. Removing the incident inter-block edges left of P for each vertex of P removes at least one edge of each simple ring in S.

Let R be a simple ring in S. Assume that $b > 1$ inter-block edges $\{e_1, \ldots, e_b\} \subseteq E_{\text{inter}}$ of R were removed. Then for each $e_i = (w, v_j)$ the vertex w is left of P and v_j is in P. Since R traverses inter-block edges only in their direction, R cannot leave P to the left. Thus R enters P b times from the left and has to leave it b times to the right (to be able to enter it again). Then $\text{wind}(R) = b > 1$, which contradicts Lemma 3. □

4.2.3 Compacting

We compact the hierarchical block graph S' in a different way as in [8] by Algorithm 5: The array X stores the last used x-coordinate on level i at $X[i]$ (lines 2–6). We place each block, which is a source in the acyclic block graph, on an imaginary zero line, treat all other blocks in topological order, and move them as much to the left as possible, preserving unit distance (lines 7–11). On a high level this corresponds to a leftmost longest path compaction of the block graph with the blocks as super vertices. Afterwards, we fix all sinks at their positions, consider all other blocks in reversed topological order, and move them as much to the right as possible (lines 12–18). This corresponds to a rightmost longest path compaction preserving the x-coordinates of the sinks. In order to place a block as close as possible to the already placed ones, we traverse its levels (lines 8 and 15). After the compaction each block and therefore each vertex v in S' has an assigned x'-coordinate.

4.2.4 Determining the Slope

Let $e = (u, v) \in R$ be a removed inter-block edge. The width of the widest simple ring of S through e is $x'(u) - x'(v) + 1$, where x' are the x-coordinates in the compacted drawing. Considering all removed inter-block edges and computing the maximum value gives the width of the widest simple ring $\text{width}(S)$ in S (line 19 of Algorithm 5).

See Fig. 10 for an example of an SCC S with $\text{wind}(S) = 1$ and $k = 6$ levels. The dotted line in Fig. 10(a) cuts six inter-block edges. Figure 10(b) shows the

Algorithm 5: compactBlocks

Input: A hierarchical k'-level block graph $S = (V, E_{\text{intra}} \dot\cup E_{\text{inter}}, \phi')$ and the set of removed inter-block edges E_I

Output: The width of S and the values $x'(v)$ set for each $v \in V$

```
1  Let B be the list of blocks of S in topological order from the left
2  l_min ← min_{v∈V}{φ'(v)}
3  l_max ← max_{v∈V}{φ'(v)}
4  Let X be an array of size k' with indices [l_min, ..., l_max]
5  for l_min ≤ i ≤ l_max do
6      X[i] ← −1
7  for B ∈ B do in topological order
8      x ← max_{l∈levels(B)} X[l]
9      foreach v ∈ V(B) do
10         x'(v) ← x + 1
11         X[φ'(v)] ← x + 1
12 for B ∈ B do in reversed topological order
13     if B has outgoing inter-block edges then
14         Let L be the set of levels of B with outgoing inter-block edges
15         x ← min_{l∈L} X[l]
16         foreach v ∈ V(B) do
17             x'(v) ← x − 1
18             X[φ'(v)] ← x − 1
19 w ← max_{e=(u,v)∈E_I} (x'(u) − x'(v) + 1)
20 return w
```

resulting compacted hierarchical block graph using the leveling ϕ'. The widths of the widest rings through each of the six cut edges $e_1, \ldots, e_6$ are 5, 5, 7, 10, 10, 10. Consequently, $\text{width}(S) = 10$. This example shows the necessity to traverse intra-block edges in two directions. Otherwise, the widest ring would have width 9 as the blocks B and C could not be in the same ring. Skewing the drawing by the slope $\frac{-1 \cdot 6}{10}$ results in Fig. 10(c). Using the modulo operation for the y-coordinates gives the final intermediate drawing in Fig. 10(d). Algorithm 6 computes the coordinates of the intermediate drawing.

Algorithm 6: skew

Input: A hierarchical block graph $S = (V, E_{\text{intra}} \dot\cup E_{\text{inter}}, \phi')$ and the slope s

Output: A skewed cyclic block graph with $x(v)$ and $y(v)$ set for each $v \in V$

```
1 for v ∈ V do
2     x(v) = x'(v) + y'(v)/s
3     y(v) = ((φ'(v) − 1) mod k) + 1    // = φ(v)
4 S' ← (V, E_intra ∪̇ E_inter, y)
5 return S'
```

Theorem 2 *For an SCC S of a block graph of a cyclic k-level graph let the intermediate drawing of S use the coordinates $x(v) = x'(v) - (\text{width}(S)/(\text{wind}(S) \cdot k)) \cdot \phi'(v)$ and $y(v) = ((\phi'(v) - 1) \bmod k) + 1 = \phi(v)$ for each vertex v in S. Then the drawing is slope aligned with slope* $\text{slope}(S) = -(\text{wind}(S) \cdot k)/\text{width}(S)$.

Proof: In the compacted drawing of the hierarchical block graph all blocks are drawn vertically. Skewing the drawing does not change the ϕ'-coordinates and results in the new x-coordinates $x(v) = x'(v) + \phi'(v)/\text{slope}(S)$. Now all edges have a slope with value $\text{slope}(S)$. Using the y-coordinates $y(v) = ((\phi'(v) - 1) \bmod k) + 1 = \phi(v)$ does not affect the slope of the edges but results in the same y-coordinates of all vertices on the same level again. Let u and v be two consecutive vertices on the same level. Let u be left of v in S as defined by the inter-block edge (u, v). If Algorithm 4 did not cut (u, v), then u and v have the same ϕ'-coordinate in the compacted drawing and u is the left neighbor of v with at least unit distance between them. This does not change in the skewed or resulting intermediate drawing. If (u, v) was cut, then $\phi'(v) = \phi'(u) - k \cdot \text{wind}(S)$. Then there is a simple block path P from v to u, since we are compacting an SCC. P cannot have been cut, as otherwise P and (u, v) form a simple ring that would have been cut twice. The ring formed by P and (u, v) is at most $\text{width}(S)$ wide and thus $x'(v) \geq x'(u) - (\text{width}(S) - 1)$. After skewing the drawing, $x(v) \geq x(u) + 1$ holds. Therefore, u is left of v and the two vertices are at least one unit apart. As a result, all consecutive vertices and so all vertices on the same level are separated at least by unit distance and they have kept their initial order. □

4.2.5 Compaction of all Compacted Strongly Connected Components

Our next step is a global compaction of the set of compacted complex and simple SCCs, which yields slope uniform drawings. First, we show that all SCCs are separate.

Lemma 6 *A drawing of a cyclic k-level graph respecting the vertex order places all vertices of an SCC consecutively on their level.*

Proof: Let u and w be two vertices of an SCC S on some level l with u left of w. Suppose there is a vertex v on level l with $u < v < w$ which does not belong to S. Then there is a block path from w to u, and there are horizontal paths from u to v and v to w using inter-block edges only. Therefore, v is in a ring containing u and w and belongs to S, which is a contradiction. □

Hence, SCCs cannot interleave. We interpret the SCCs as super vertices and run a longest path algorithm on the resulting DAG. We then compact the SCCs as we compact the blocks of a hierarchical block graph. One simple problem remains: The leftmost (rightmost) vertices of each level of the SCC may have different x-coordinates. Thus the SCC has no straight vertical left and right borders. To compute the shift of one SCC we simply traverse all vertices of the SCC. Algorithm 7 gives the details of this compaction step. It places the SCCs next to each other and as close as possible.

4.3 Balancing

In this phase (see Algorithm 8) the four results are balanced by computing one x-coordinate for each vertex from the four x-coordinates computed by the four preceding runs. We differ from the algorithm of Brandes and Köpf [8] and do not use the average median of the four x-coordinates for each vertex, since in the cyclic case this can induce additional bends. The reason is that the median of lines with different slopes changes at their crossings, i. e., it is a non-linear function. Hence, we use the average of all four x-coordinates for each vertex. The result is not integral, but still of bounded precision.

Lemma 7 *Algorithm 8, averaging over the x-coordinates of the four runs to determine the final x-coordinate for each vertex, constructs a so called* weakly slope aligned *drawing. This means, it preserves the order of the vertices on each level. It guarantees at most two bends per edge occurring only at the topmost and bottommost dummy vertices. It does not add bends to the edges of subgraphs, which belong to an SCC in all four runs. There are additional bends, since the blocks of the four runs may differ. Finally, it preserves at least a horizontal unit distance between the vertices and has a discrete resolution.*

Proof: Each drawing of the four runs has the same order and at least unit distance on each level. Thus both properties hold for the average of the four runs. Let $e_1 = (u, v)$ and $e_2 = (v, w)$ be two intra-block edges in all four runs.

Algorithm 7: compact

Input: The set $\mathcal{S}$ of compacted SCCs of a cyclic k-level graph
Output: The values $x(v)$ set for each vertex v of each SCC of $\mathcal{S}$

Let $\mathcal{S}$ be the list of SCCs in topological order from the left
Let X be an array of size k with indices $[1, \ldots, k]$
for $1 \leq i \leq k$ **do**
    $X[i] \leftarrow -1$
for $S \in \mathcal{S}$ **do** in topological order
    $\delta_x \leftarrow \min_{v \in V(S)}(x'(v) - X[\phi(v)])$
    foreach $v \in V(S)$ **do**
        $x(v) \leftarrow x'(v) - \delta_x + 1$
        if $x(v) > X[\phi(v)]$ **then**
            $X[\phi(v)] \leftarrow x(v)$
for $S \in \mathcal{S}$ **do** in reversed topological order
    if *S has outgoing inter-block edges into another SCC* **then**
        Let M be the set of vertices of S with outgoing inter-block edges into another SCC
        $\delta_x \leftarrow \min_{v \in M}(X[\phi(v)] - x(v))$
        foreach $v \in V(S)$ **do**
            $x(v) \leftarrow x(v) + \delta_x - 1$
            if $x(v) < X[\phi(v)]$ **then**
                $X[\phi(v)] \leftarrow x(v)$

Algorithm 8: balance

Input: The set $\mathcal{P}$ of the results of the four runs of $G' = (V', E', \phi', <)$
Output: $x(v)$ and $y(v)$ for each vertex $v \in V'$

Let $x_i(v)$ be the x-coordinate of the vertex v in the i-th run $(1 \leq i \leq 4)$
foreach $v \in V'$ **do**
    $x(v) \leftarrow (x_1(v) + x_2(v) + x_3(v) + x_4(v))/4$
    $y(v) \leftarrow \phi'(v)$

Then e_1 and e_2 are drawn with the same slope in each of the four drawings. Thus they have identical slopes in the balanced final drawing, i. e., there is no bend at v. As all dummy vertices of a long edge belong to one block in each drawing, the argument holds for each of its non-extremal dummy vertices.

As in each run the x-coordinate of a vertex is an integral multiple of $\frac{1}{k}$ (Theorem 2), the average of them is an integral multiple of $\frac{1}{4k}$. Thus we have a discrete resolution in dependency of $|V|$. □

Note that some vertices may belong to a block of a complex SCC in one run although they do not belong to a complex SCC or even one block in another run. Consequently, balancing can lead to more slopes than in any of the four runs. See Fig. 8 for two different block graphs of two runs of the same graph.

If similar slopes are more important than balancing, we compute only one downwards *balanced run* with a modified block building phase. We start with the median vertex of the upper level and align it with its median successor. Whenever one of the two medians is not unique, we choose arbitrarily. From this vertex we traverse to the left (right) to align the remaining vertices on the level trying the right (left) median first. However, this results in balanced outgoing edges only. We compact unsymmetrically to the left first as described in Sect. 4.2.3.

5 Algorithm Analysis

Next we establish upper and lower bounds on the width and the area of the resulting drawings and of the running time of the algorithm.

Theorem 3 *Let $G = (V, E, \phi, <)$ be a (not necessarily proper) ordered cyclic k-level graph. Algorithm 1 constructs slope aligned drawings of G. The intermediate and the 3D drawing have a width of $\mathcal{O}((|V| + |E|)^2)$ and an area of $\mathcal{O}((|V| + |E|)^2 \cdot k)$. The 2D drawing has a width and a height of $\mathcal{O}((|V| + |E|)^2)$ and an area of $\mathcal{O}((|V| + |E|)^4)$.*

Proof: Let $\mathcal{S} = \{S_1, \ldots, S_r\}$ be the set of SCCs of G. Let β_i be the number of blocks in S_i and ν_i be the number of (dummy) vertices in S_i. The width of the compacted drawing of S_i is at most β_i. The height of the drawing is at most the sum of the height of all blocks and is therefore bounded by ν_i. Skewing this drawing by a slope of $-\text{wind}(S_i) \cdot k/\text{width}(S_i)$ adds at most $(\nu_i \cdot \beta_i)/k$ to the width. As $\nu_i \leq \beta_i \cdot k$, the width of the drawing of S_i is in $\mathcal{O}(\beta_i^2)$. The width of the drawing of G is $\mathcal{O}((|V| + |E|)^2)$, since it is at most the sum of the widths of the drawings of all SCCs and $\sum_{i=1}^{r} \beta_i^2 \leq (\sum_{i=1}^{r} \beta_i)^2 \leq (|V| + |E|)^2$.

The area is in $\mathcal{O}((|V| + |E|)^2 \cdot k)$, since the height is k. The height and width of the 2D drawing is twice the width of the intermediate drawing, which results in an area of $\mathcal{O}((|V| + |E|)^4)$.

The slope alignment holds by Theorem 2 and using one balanced run. □

Note that the width of the drawing reduces to $\mathcal{O}(|V| + |E|)$ if there are no complex SCCs in the graph. This reduces the area of the intermediate and 3D

drawings to $\mathcal{O}((|V|+|E|)\cdot k)$ and of the 2D drawing to $\mathcal{O}((|V|+|E|)^2)$. Complex SCCs can always be avoided at the cost of crossings and balance by using our global sifting heuristic and a modified block building algorithm as we show in the next section.

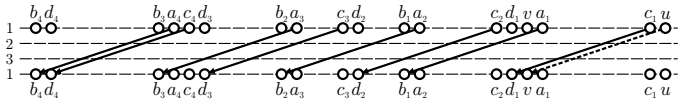
Although the width of the produced drawings can be quadratic, benchmarks [9] have shown that the average width is much smaller. Note that in the degenerate case of just one level, the normalization step (Algorithm 3) removes all intra-block edges from the block graph. Hence, the vertices are compacted without any constraints and the width is exactly $|V|$.

Next, we prove the lower bound of the width of such drawings. For 2D and 3D drawings we use δ_x instead of unit distance for consecutive vertices as described in Sect. 2. For simplicity we use the term unit distance nevertheless. We now show that there are block graphs such that any slope aligned drawing has quadratic width.

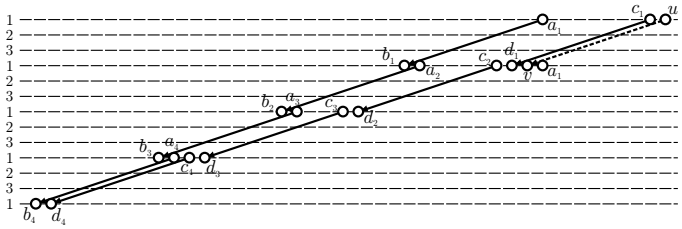
Theorem 4 *For each number of vertices $|V|$ and levels $k > 1$ there exists an ordered cyclic k-level graph $G = (V, E, \phi, <)$ such that each slope aligned intermediate drawing of G has a width of $\Omega(|V|^2)$. The area of such a drawing is in $\Omega(|V|^2 \cdot k)$. For slope aligned 3D drawings of G the same bounds hold. Each slope aligned 2D drawing of G has a width and height of $\Omega(|V|^2)$ and an area of $\Omega(|V|^4)$.*

Proof: For the proof we first consider intermediate drawings. We construct a family of ordered cyclic k-level graphs $\{C_{n,k} = (V_{n,k}, E_{n,k}, \phi_{n,k}, <_{n,k})\}_{n,k\in\mathbb{N}}$ which achieve the stated bounds. For a concise description the graphs are first simplified. $V_{n,k}$ consists of $4n$ vertices a_i, b_i, c_i, d_i for each $1 \leq i \leq n$ and 2 additional vertices u, v, which all lie on level 1. The left-to-right order of the vertices is $b_n <_{n,k} d_n <_{n,k} b_{n-1} <_{n,k} a_n <_{n,k} c_n <_{n,k} d_{n-1} <_{n,k} \ldots <_{n,k} b_1 <_{n,k} a_2 <_{n,k} c_2 <_{n,k} d_1 <_{n,k} v <_{n,k} a_1 <_{n,k} c_1 <_{n,k} u$. $E_{n,k}$ has $2n$ directed edges (a_i, b_i) and (c_i, d_i) for each $1 \leq i \leq n$ and the additional edge (u, v). The order of the (dummy) vertices on levels 2 to k is completely defined by the unique planar routing of the straight-line edges. See Fig. 11 for possible drawings of $C_{4,3}$.

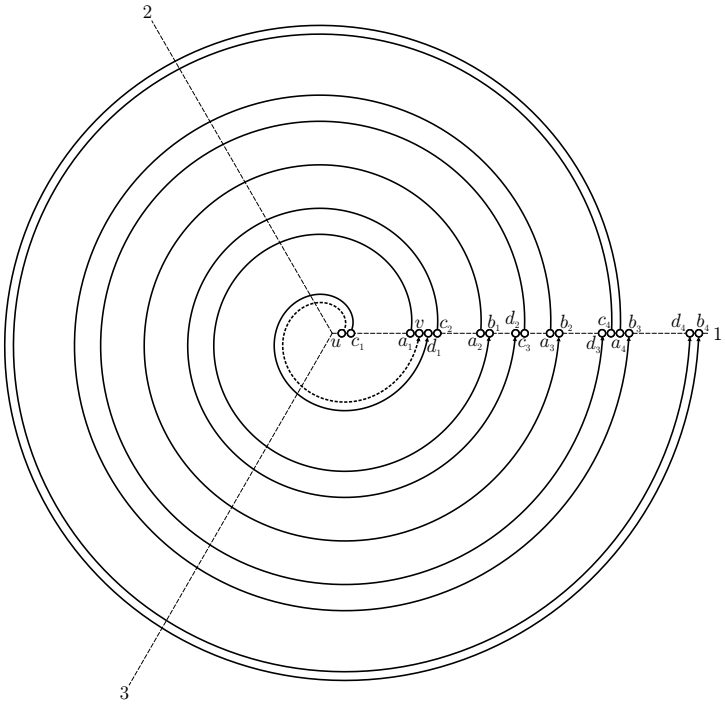
As there are no crossings in the ordered cyclic level graph and all original vertices have degree 1, the block graph of $C_{n,k}$ returned by Algorithm 2 contains all segments of $C_{n,k}$. However, the normalization step in Algorithm 3 removes some segments of the block graph. This can be avoided by a slight modification of the graphs $C_{n,k}$, splitting each edge into two non-adjacent edges and introducing two new vertices on level 2. We suppress this detail for now. Each of the $2n + 1$ edges represents exactly one block consisting of k segments. Thus we identify $C_{n,k}$ with its cyclic block graph. $C_{n,k}$ consists of a single SCC, as the vertices $a_1, b_1, a_2, b_2, \ldots a_n, b_n, d_n, c_n, \ldots, d_1, c_1, u, v, a_1$ form a ring R with $\text{wind}(R) = 1$. For a slope aligned drawing of $C_{n,k}$ the only degree of freedom is the slope of all blocks and the distances of each pair of consecutive vertices on the same level. Consider an arbitrary slope aligned intermediate drawing Γ with slope s. A complete traversal of R from a_1 ends at a_1 (see Fig. 11(b)). Hence,



(a) Quadratic width in the intermediate drawing



(b) Quadratic width in the skewed drawing (with duplicated vertex a_1)



(c) Quadratic width and height in the cyclic 2D drawing

Figure 11: Graph $C_{4,3}$

the sum of the changes in the x-coordinates of the vertices is 0 while traversing R. Note that traversing one segment of slope s results in the change in the x-coordinate by $\pm\frac{1}{s}$ depending on the direction of the traversal. While R is traversed from a_1 to b_n n edges, i. e., kn segments, with slope s are traversed in their direction. This leads to a change in the x-coordinate by $kn \cdot \frac{1}{s}$. Traversing R from d_n to c_1 uses kn segments with slope s against their direction, which leads to a change in the x-coordinate by $-kn \cdot \frac{1}{s}$. Finally, (u, v) consists of k segments with slope s and is traversed in its direction inducing a change of $k \cdot \frac{1}{s}$. Furthermore, while traversing R each of the $2n + 1$ blocks is left once to the consecutive right block. Each time this adds at least a unit distance to the change in the x-coordinate, i. e., $\delta \geq 2n + 1$ in total. The change in the x-coordinate is therefore $0 = kn/s - kn/s + k/s + \delta$. Hence, the (negative) slope is at least $s = -k/\delta \geq -k/(2n + 1)$.

Consider the difference in the x-coordinates of the vertices d_n and c_1 in Γ. The path between them consists of kn segments and $n - 1$ horizontal distances. As the segments have a slope of at least $-k/(2n + 1)$ and consecutive vertices on the same level have at least unit distance, the absolute difference in the x-coordinates of d_n and c_1 is at least $-kn/s + (n - 1) \cdot 1 \geq kn \cdot (2n + 1)/k + (n - 1) \in \Omega(n^2)$. As the number of vertices $|V_{n,k}|$ is linear in n, each slope aligned intermediate drawing of $C_{n,k}$ has a width of $\Omega(|V_{n,k}|^2)$.

If we split each edge (x, y) of $C_{n,k}$ into two edges (x, x') and (y', y) with $\phi(x') = \phi(y') = 2$, all edges of the graph have a span of at most $k - 1$. Hence, the normalization step (Algorithm 3) does not remove any segment. If the ring R traverses (x, y) in its direction we set $x' < y'$ and $y' < x'$, otherwise. Thus R contains the same number of segments as before. However, the number of blocks has doubled. Then the blocks have to be skewed even more since the slope is halved and the width is still quadratic.

The leftmost vertex b_n and the rightmost vertex u of level 1 differ by $\Omega(|V_{n,k}|^2)$ in their x-coordinates. Since the edge (a_n, b_n) has slope s, the x-coordinates of a_n and b_n differ by at most $|k/s| \leq k/(k/(2n + 1)) \in \mathcal{O}(n)$. For the edge (u, v) the same argument holds. The leftmost (rightmost) vertices of the levels 2 to k are dummy vertices of the edge (a_n, b_n) ((u, v)). Thus each leftmost and rightmost dummy vertex differs in their x-coordinates by at least $\Omega(|V_{n,k}|^2)$. Furthermore, the difference of the x-coordinates of the leftmost and the rightmost intersection of each horizontal straight line with the drawing is in $\Omega(|V_{n,k}|^2)$. Note that splitting each edge even increases the distance between the leftmost and rightmost vertex on each level.

For slope aligned 3D drawings the same arguments hold. For 2D drawings assume for contradiction that there is a slope aligned 2D drawing of $C_{n,k}$ with width or height in $o(|V_{n,k}|^2)$. Using the reverse transformation of intermediate drawings to 2D drawings of Sect. 2 leads to a slope aligned intermediate drawing where a horizontal straight line with a difference in extremal intersections of $o(|V_{n,k}|^2)$ can be found at least once. This is a contradiction. Hence, the width and height of the slope aligned 2D drawing are in $\Omega(|V_{n,k}|^2)$ and the area is in $\Omega(|V_{n,k}|^4)$. □

Since the graphs of Theorem 4 used to prove the lower bound are planar, we conclude:

Corollary 1 *Let $G = (V, E, \phi, <)$ be a (not necessarily proper) ordered planar cyclic k-level graph. The bounds on the width and the area of slope aligned intermediate drawings of G are $\Theta((|V|+|E|)^2)$ and $\Theta((|V|+|E|)^2 \cdot k)$. For slope aligned 3D drawings of G the same bounds hold. Each slope aligned 2D drawing of G has a width and a height of $\Theta((|V|+|E|)^2)$ and an area of $\Theta((|V|+|E|)^4)$.*

In addition, just three slopes suffice for a drawing within the shown area bounds.

Corollary 2 *Let $G = (V, E, \phi, <)$ be a (not necessarily proper) ordered cyclic k-level graph. Then there exists a slope s such that G has a uniformly slope aligned intermediate drawing with slope s and width $\mathcal{O}((|V| + |E|)^2)$.*

Proof: Let Γ' be a slope aligned intermediate drawing of G computed with Algorithm 1 with one balanced run. Then Γ' has a width of $\mathcal{O}((|V| + |E|)^2)$. Let $\mathcal{S}$ be the set of all complex SCCs of the block graph of G and let s be the minimum of the absolute slopes of the segments in each SCC $S \in \mathcal{S}$. For each $S \in \mathcal{S}$ we construct a new drawing by multiplying the x-coordinates by $|s'|/s$, where s' is the uniform slope of S. Then all segments in S have a uniform slope of $\pm s$. Compacting these new drawings of the complex SCCs together with the unchanged drawings of the simple SCCs results in a drawing Γ where all slopes are in $\{-s, \infty, s\}$. As the horizontal distances between the vertices are increased, the minimum of one unit is preserved.

The minimum absolute slope used in Algorithm 1 for skewing an SCC S is $k/\text{width}(S) \geq k/(|V|+|E|)$, see the proof of Theorem 3. Thus $s \geq k/(|V|+|E|)$. Traversing a segment with slope s results in the change in the x-coordinates of at most $(|V|+|E|)/k$. Since each block consists of at most k segments, its traversal results in a change of at most $(|V| + |E|)$. Placing all $\mathcal{O}(|V| + |E|)$ blocks side by side without compacting results in a drawing Γ with width $\mathcal{O}((|V| + |E|)^2)$. □

Note that the construction in the proof of Corollary 2 enforces non-integral x-coordinates for the vertices. Furthermore, the segments in the drawing of G which are not in the block graph have different slopes. The runtime of the algorithm in total can be bounded as follows.

Theorem 5 *The layout algorithm (Algorithm 1) has time complexity $\mathcal{O}(|V'| + |E'|)$ for a proper ordered cyclic k-level graph $G' = (V', E', \phi', <')$.*

Proof: Flipping the input graph horizontally and/or vertically can be done in $\mathcal{O}(|V'|+|E'|)$ time. Building the cyclic block graph consists of two parts: Marking all type 1 conflicts takes $\mathcal{O}(\deg^-(v))$ time for one vertex and $\mathcal{O}(|V'|+|E'|)$ in total. Aligning each (dummy) vertex with a median neighbor is done in $\mathcal{O}(|V'|)$. The size of the cyclic block graph is $\mathcal{O}(|V'|)$ as it contains $\mathcal{O}(|V'|)$ (dummy) vertices, intra-block edges, and inter-block edges. Normalizing long blocks results

in traversing each block completely and takes $\mathcal{O}(|V'|)$ time. Computing the strongly connected components of the cyclic block graph can be done in time $\mathcal{O}(|V'|)$. Cutting an SCC, assigning new levels ϕ', compacting, and skewing an SCC has a time complexity linear in the size of the SCC, which adds up to $\mathcal{O}(|V'|)$ for all SCCs. Finally, compacting G' is done in linear time as well. All these steps are carried out four times. Finally, balancing the four results takes $\mathcal{O}(|V'|)$ time. □

6 Avoiding Cyclic Dependencies

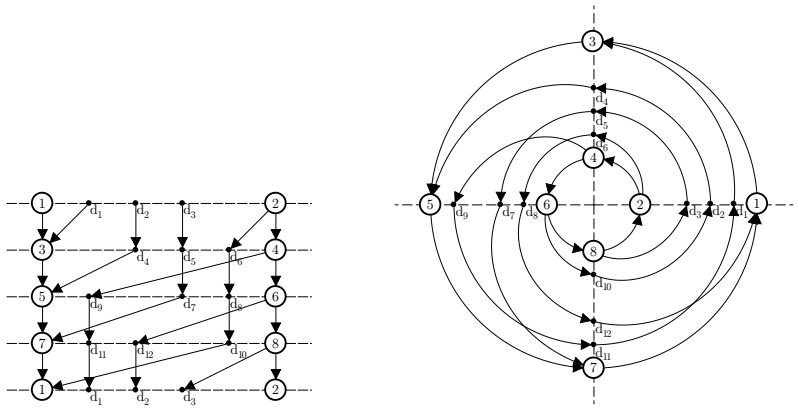
Cyclic dependencies in the left-to-right order of cyclic level graphs force skewed inner segments. Skewing gives symmetric layouts at the expense of the width of the drawing. In the coordinate assignment phase it is too late to avoid these dependencies as they are generated during the crossing reduction phase. Fig. 6 shows a cyclic level graph which inevitably has cyclic dependencies when the optimal permutations on all levels are used. An optimal cyclic crossing reduction [19] or a cyclic level planarity testing and embedding algorithm [6] finds these permutations and therefore produces cyclic dependencies.

One option is using the global sifting algorithm [3] for cyclic crossing reduction. There each original vertex and each chain of inner segments has a distinct x-coordinate. These subgraphs are called blocks as well, but they are smaller than the ones used here. Using these x-coordinates results in an intermediate drawing where all inner segments are vertically aligned. However, drawing all inner segments vertically is impossible if there are cyclic dependencies. Thus there are no cyclic dependencies among these blocks. Fig. 12 shows a respective drawing of G_2 from Fig. 6 with vertical inner segments. As illustrated the aesthetic criterion of vertical inner segments generally costs additional crossings, which do not occur in the drawing with skewed blocks. Since the coordinate assignment should leave the vertex orders unchanged, a crossing reduction is desirable which supports vertical blocks.

Our cyclic coordinate assignment phase (as well as [8]) uses larger blocks than the global sifting heuristic [3]. It tries to assemble as many (dummy) vertices as possible in a block which may result in cyclic dependencies.

When applying the global sifting heuristic for crossing reduction, there are two options which blocks are used in our coordinate assignment algorithm. The *minimal blocks* from the crossing reduction avoid cyclic dependencies and result in vertical inner segments and a linear width of the drawing. But the drawings are unbalanced as only dummy vertices are aligned. Using the *maximal blocks* described in this paper places vertices balanced with respect to their neighbors. However, there may be cyclic dependencies which cause skewing inner segments and potentially quadratic width.

Consider the intermediate drawings of the cyclic 4-level graph G_3 of Fig. 13 and Fig. 14. G_3 consists of four edges each spanning three levels. With maximal blocks the graph has cyclic dependencies, e. g., $x(1) < x(d_1) = x(3) < x(d_3) = x(5) < x(d_5) = x(d_7) = x(1)$, see Fig. 14(a). The vertices are aligned, but the



(a) Intermediate drawing (b) Cyclic drawing

Figure 12: Graph G_2 from Fig. 6 with vertical inner segments at the expense of crossings

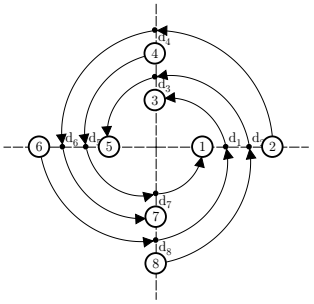
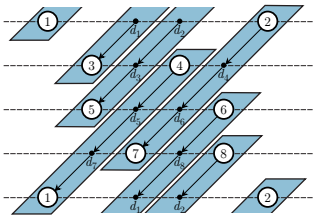


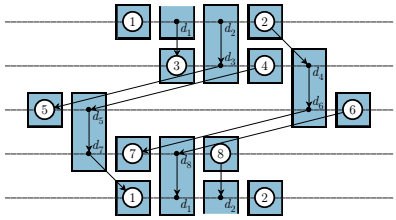
Figure 13: Ordered graph G_3

edges must be skewed, and the drawing may have a quadratic width. However, there are no such rings when using the minimal blocks of the global sifting algorithm [3]. Fig. 14(b) shows such a drawing with vertical inner segments and fewer aligned vertices.

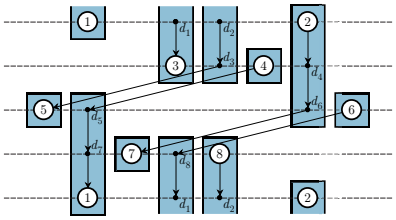
Obviously, using smaller blocks is only reasonable if larger ones would cause a cyclic dependency. Therefore, we start with the maximal blocks and only split them into smaller ones if they imply a cyclic dependency. Reconsider Fig. 14(a). When using maximal blocks all eight outer segments are used for aligning. However, removing four of them already destroys all cyclic dependencies. One possible result is shown in Fig. 14(c): The four outer segments in the blocks result in the alignment of the vertices 1 and 2 in contrast to Fig. 14(b). By coincidence, the vertices 3 and 8 were already aligned in Fig. 14(b).



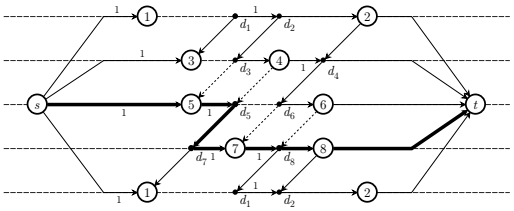
(a) Intermediate drawing with maximal blocks



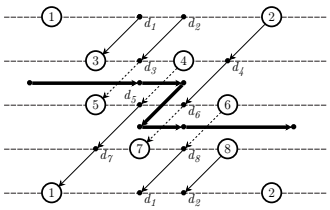
(b) Intermediate drawing with minimal blocks



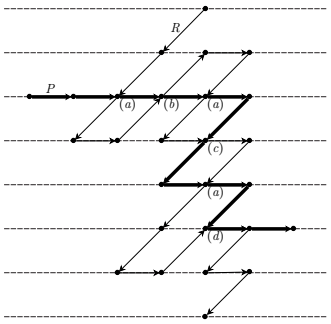
(c) Intermediate drawing with largest blocks without cyclic dependencies



(d) Outer segments graph



(e) Interpretation as dual graph



(f) Crossings of paths and rings

Figure 14: The deletion of a minimal set of intra-block edges for the ordered graph G_3 from Fig. 13

For our new goal we must find a minimal set of intra-block edges of outer segments whose removal results in an acyclic block graph. Finding such a minimum set in an arbitrary graph corresponds to the $\mathcal{NP}$-hard feedback arc set problem [18]. But a cyclic block graph has a very regular structure: Each vertex has at most two incoming and two outgoing edges forming a grid with edges pointing to the right and downwards. Furthermore, it is plane and all rings wrap around the center in the same direction. Note that the feedback arc set problem in planar graphs can be solved in polynomial time [7]. For the special case of removing all clockwise or counter-clockwise cycles in a fixed embedding an $\mathcal{O}(|V| \log |V|)$ time algorithm has been established [28].

However, we only want to remove intra-block edges of outer segments to destroy all rings. Furthermore, rings can traverse intra-block edges in both directions. In this special case, we can find the minimal number of such edges in linear time. The idea is similar to cutting SCCs. However, this time we cut intra-block edges instead of inter-block edges and we cut as few as possible. Each SCC (with maximal blocks) is treated separately.

Definition 5 *Let S be an SCC with* $\mathrm{wind}(S) = 1$. *Define the* outer segments graph *S' of S to consist of the vertices of S and of two new vertices s and t. The edges of S' have length* 0 *or* 1.

There is a directed edge of length 0 *from each rightmost vertex of S to t. S' contains all intra-block edges of S with assigned length* 0.

Let v be a vertex with an incoming inter-block edge $e = (u, v) \in E_{\text{inter}}$. If v is the topmost vertex of its block, then S' contains e with length 0. *Otherwise, v has an incoming intra-block edge e'. If e' represents an outer segment of the cyclic level graph, then S' contains e with length* 1, *else e' represents an inner segment and S' does not contain e.*

Let v be a leftmost vertex on its level. If v is the topmost vertex of its block, then S' contains the edge (s, v) with length 0. *Otherwise, v has an incoming intra-block edge e'. If e' represents an outer segment of the cyclic level graph, then S' contains the edge (s, v) with length* 1. *Otherwise, e' represents an inner segment and S' does not contain the edge (s, v).*

We now prove that the length of the shortest directed path from s to t in the outer segments graph is the number of intra-block edges whose deletion destroys all cyclic dependencies. We remove the incoming intra-block edge of v of each used edge (u, v) with length 1. Fig. 14(d) shows the resulting outer segments graph for the graph in Fig. 14(a), where length 0 is not indicated. Each vertex with an incoming outer segment has an incoming edge from the left with length 1. The shortest path from s to t is drawn in bold. The vertices 5, d_5, 7 and d_8 are entered from the left and have an incoming outer segment. These four outer segments are dashed and are removed in Fig. 14(c). Note that the drawing of the outer segments graph in Fig. 14(d) contains duplicates of horizontal edges on the first and last levels, since the corresponding vertices are drawn twice. The graph itself has the same cyclic structure as the cyclic block graph.

The intuition behind the outer segments graph is as follows: An outer segments graph is (apart from s and t) a subgraph of the cyclic block graph. Both

have a grid structure: Each vertex has at most one incoming edge from above and one from the left, one outgoing edge downwards, and one outgoing edge to the right. A path from s to t in the outer segments graph is only allowed to traverse in edge direction, i. e., downwards and to the right. A ring in the cyclic block graph can use intra-block edges in both directions and inter-block edges in their direction only. Hence, a ring in the cyclic block graph traverses upwards, downwards and to the right.

Intuitively, we remove the incoming intra-block edge each time the path from s to t in the outer segments graph enters a vertex from the left. We can only do so if the edge from the left exists, which is the case if the intra-block edge from above represents an outer segment. As these edges are the only ones with length 1, their number is minimized in a shortest path from s to t.

This method has some similarities to [28] which uses the dual graph of the input graph to solve the feedback arc set problem in planar graphs. Due to the regular grid structure of the block graph the outer segments graph resembles the dual graph of the block graph. Each vertex v of the outer segments graph can be interpreted as the vertex of the dual graph belonging to the face of the grid above the right outgoing edge of v. In Fig. 14(e) the path of Fig. 14(d) is shifted to represent a path in the dual graph. Now the path crosses exactly the outer segments we remove. We use the dual graph as an intuition only as the proofs benefit from having the same vertex set for the block graph and the outer segments graph.

Lemma 8 *Let S be a complex SCC with* $\mathrm{wind}(S) = 1$ *of a cyclic block graph using maximal blocks such that there is no ring when using minimal blocks. Let P be a path from s to t in the outer segments graph. Further, let Q be the set of vertices of P, which are entered from the left by P. Then, removing the incoming intra-block edge of each vertex $q \in Q$ from S results in an acyclic block graph.*

Proof: Let R be a simple ring in S. Then $\mathrm{wind}(S) = \mathrm{wind}(R) = 1$ holds. Since R is a cycle in the cyclic block graph, it splits the block graph into a left and a right part. Since P is a path from a leftmost to a rightmost vertex, R and P have at least one common vertex. Let v be the first common vertex on P. Hence, the predecessor of v in P lies in the left part of the block graph. Note that P can only traverse to the right and downwards and R can only traverse to the right, upwards, and downwards. P and R cannot enter v using the same edge since then v is not the first common vertex. Fig. 14(f) contains the remaining four cases. If P enters v from above and R from the left (case (c)) or from below (case (d)) or if P enters from the left and R from below (case (b)), then the path P has already been in the right part of the block graph. Hence, this cannot be the first crossing. The remaining case (case (a)) is that P enters from the left and R from above. Then the incoming intra-block edge of v is one of the removed edges and R does not exist in the new block graph. □

Lemma 9 *Let S be a complex SCC of a cyclic block graph using maximal blocks such that there is no ring when using minimal blocks. Let* $\mathrm{wind}(S) = 1$. *Let*

M be a minimal set in terms of inclusion of intra-block edges, whose removal results in an acyclic block graph. Then there exists a path in the outer segments graph from s to t using $|M|$ edges of length 1.

Proof: Removing all the segments of M from S makes S acyclic. Let $L = v_1, \ldots, v_{|M|}$ be the set of lower end vertices of the outer segments in M sorted in topological order of their minimal blocks. We search for a path from s to t by traversing all vertices of L in this order and using only edges of length 1 to enter a vertex v_i $(1 \leq i \leq |M|)$. We construct that path piecewise and backwards from each v_i.

Vertex v_1 has an incoming edge from the left, which is traversed against its direction to enter a new block B. We traverse the intra-block edges of B against their direction to the topmost vertex and repeat this procedure. Note that the topmost vertex of each block has an incoming edge from the left in the outer segments graph. Furthermore, note that all traversed edges except the first have length 0. We cannot enter a vertex a second time, since otherwise we would have traversed a ring backwards. We cannot reach a vertex v_j $(j > 1)$ as we traverse the outer segments graph in reversed topological order. Hence, we have to reach s eventually. Reversing this path gives a path from s to v_1. Starting at a vertex v_i $(i > 1)$ gives a similar result. We have to reach v_{i-1} eventually. If we reach any vertex a second time, we have found a ring. If we reach v_j $(j < i - 1)$, deleting the incoming outer segments of v_{j+1} to v_{i-1} was unnecessary. This is a contradiction to the minimality of M. For the same reason reaching s is impossible. Again, we cannot reach a vertex v_j $(j > i)$ as we traverse the outer segments graph in reversed topological order. The only remaining case is to reach v_{i-1}. A similar argument shows that a path against the edge directions from t to $v_{|M|}$ exists. Reversing and concatenating all these paths gives a path $s, \ldots, v_1, \ldots, v_2, \ldots, \ldots, \ldots, v_{|M|}, \ldots, t$ using $|M|$ edges of length 1. □

Theorem 6 *Let S be a complex SCC of a cyclic block graph using maximal blocks such that there is no ring when using minimal blocks. Let P be a shortest path from s to t in the outer segments graph. Let Q be the set of vertices of P which are entered from the left by P, and let I be the set of incoming intra-block edges of the vertices in Q. Then I is a smallest set of intra-block edges whose removal from the cyclic block graph destroys all rings.*

Proof: Lemma 8 proves that removing the incoming intra-block edges for each vertex $q \in Q$ results in a acyclic block graph. Lemma 9 shows that for each minimal set of such edges (in terms of inclusion) a corresponding path exists. Hence, the shortest path gives the smallest set of edges (in terms of cardinality) to delete. □

Finding the shortest path from s to t with edge lengths only 0 and 1 is doable in linear time with a slightly modified breadth first search.

7 Conclusion

7.1 Summary

We investigated the drawing of recurrent hierarchies and established the first coordinate assignment algorithm for cyclic level graphs. Like the algorithm of Brandes and Köpf [8] it generates at most two bends per edge, tries to draw long edges as straight parallel lines, and centers vertices among their neighbors. These are the major aesthetic criteria for such drawings. Drawing all inner segments vertically is no longer possible, as there are cyclic dependencies in the left-to-right order of the vertices. This new challenge is settled by skewing subgraphs. Hence, the symmetry of a graph can be represented in the resulting (weak) slope aligned drawing. However, the width of the drawing is quadratic. This can be avoided by combining our global sifting crossing reduction and our cyclic coordinate assignment at the cost of slightly less balanced drawings. However, all inner segments are aligned vertically then.

7.2 Discussion

We decided to base our approach on the algorithm by Brandes and Köpf, because it follows the linear segment approach and avoids the negative "spaghetti" effect with many bends per edge. Exact algorithms like Gansner et al. [16] or force-directed approaches like Sander [26] may be extendable to the cyclic case as well. However, adding restrictions to identify the vertices on the first and last levels must be done with care to preserve the feasibility of the established system.

As described in Sect. 4.3 one (balanced) run uses fewer different slopes, but each vertex is only aligned with an outgoing median. Then there is no balanced compaction. In our version we always compact first to the left and afterwards to the right. This is a bias and may induce asymmetry. The hardest challenge were cyclic dependencies. They force skewing and wider drawings. This can be avoided at the cost of more crossings and bends as discussed in Sect. 6. What is the best compromise?

There is a gap between the lower bound for the width of $\Omega(|V|^2)$ from Theorem 4 and the upper bound of $\mathcal{O}((|V| + |E|)^2)$. It would be interesting to see which bound is valid for dense graphs.

7.3 Experience

The algorithm has been implemented as part of the cyclic Sugiyama framework of the graph visualization toolkit Gravisto [5]. The resulting drawings have been compared with those from the hierarchical Sugiyama framework. Several benchmarks were run on randomly generated graphs with up to 500 vertices and 750 edges, see [9]. In summary, the cyclic framework produces drawings with a reduction of the number of bends from about 0.85 to 0.7 per edge on the average compared to the hierarchical framework. However, there is an increase of the width and of the displacements of outer segments by about 10%. The latter is

consumed by a coresponding reduction in the number of outer segments. These test indicate that the width of cyclic drawings grows only linearly with the size of the graphs, and not quadratically, which is the established worse case bound in Theorem 3.

7.4 Examples

In this section we show some example drawings produced by our implementation. Similar to the benchmarks in Sect. 7.3 these are generated with the coordinate assignment algorithm, which does not avoid cyclic dependencies within the block graph and which generates weak slope aligned drawings. Graph G_{20} has 20 original vertices and 35 edges. Fig. 15 shows a classical hierarchical drawing on 6 horizontal levels. It has 5 (dashed) back edges pointing in the wrong direction. 6 horizontal levels form 5 spaces for edge routing between them. To parallel this we used 5 levels in the cyclic case. Fig. 16 presents the final cyclic drawing of G_{20}. Fig. 17 shows the intermediate drawing of G_{20}. This is also the surface of the cylinder of the 3D drawing of G_{20}. Fig. 18 and Fig. 19 contain the same graph drawn with only one balanced run as suggested in Sect. 4.3. It is clearly visible in the intermediate drawing that this produces fewer slopes. In the cyclic drawing, however, this difference is much more difficult to recognize. All these cyclic drawings contain fewer crossings and shorter edges than the hierarchical drawing in Fig. 15.

Fig. 20 and Fig. 21 illustrate the cyclic drawing of the graph G_{24}, which consists of 24 vertices and 44 edges. Due to the 76 dummy vertices, G_{24} has 120 segments. The slope aligned drawings are again produced by only one balanced run. In Fig. 21 two separate complex SCCs of the cyclic block graph are clearly recognizable. The edges on the left are skewed with a different slope than the edges in the middle of the drawing.

Finally, Fig. 22 shows a deterministic finite automaton (DFA) using the alphabet $\Sigma = \{a, b\}$. It accepts all words with an even number of a's and whose length is a multiple of six. Clearly, the regular cyclic structure of the DFA is better represented in the cyclic drawing.

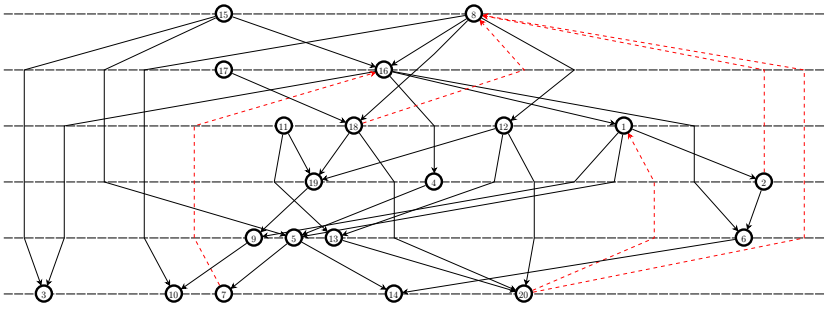


Figure 15: Hierarchical drawing of G_{20} with 20 vertices, 35 edges and 6 levels

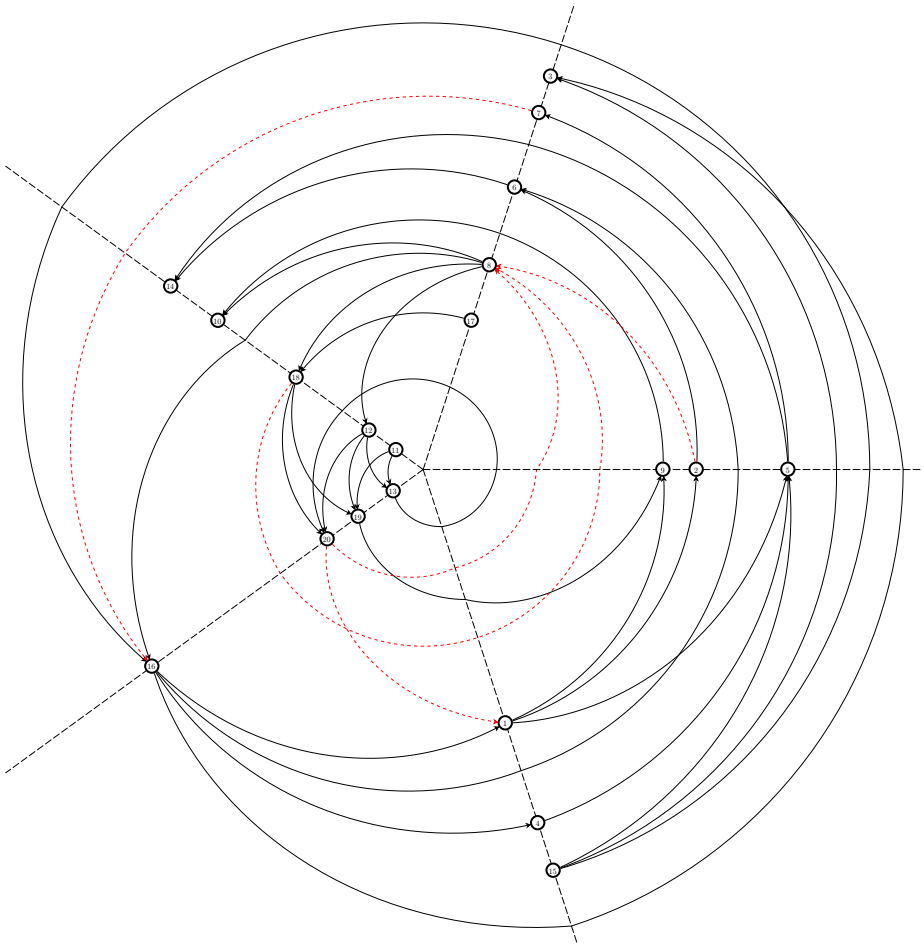


Figure 16: 2D drawing of G_{20} with 20 vertices, 35 edges and 5 levels

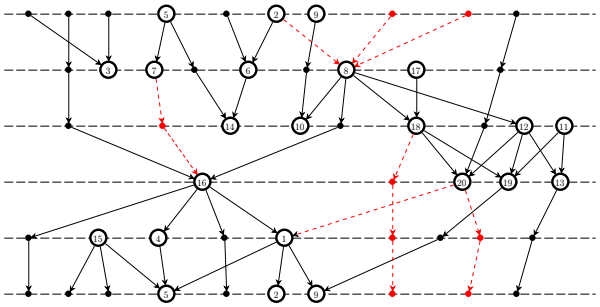


Figure 17: Intermediate drawing and surface of the 3D drawing of G_{20}

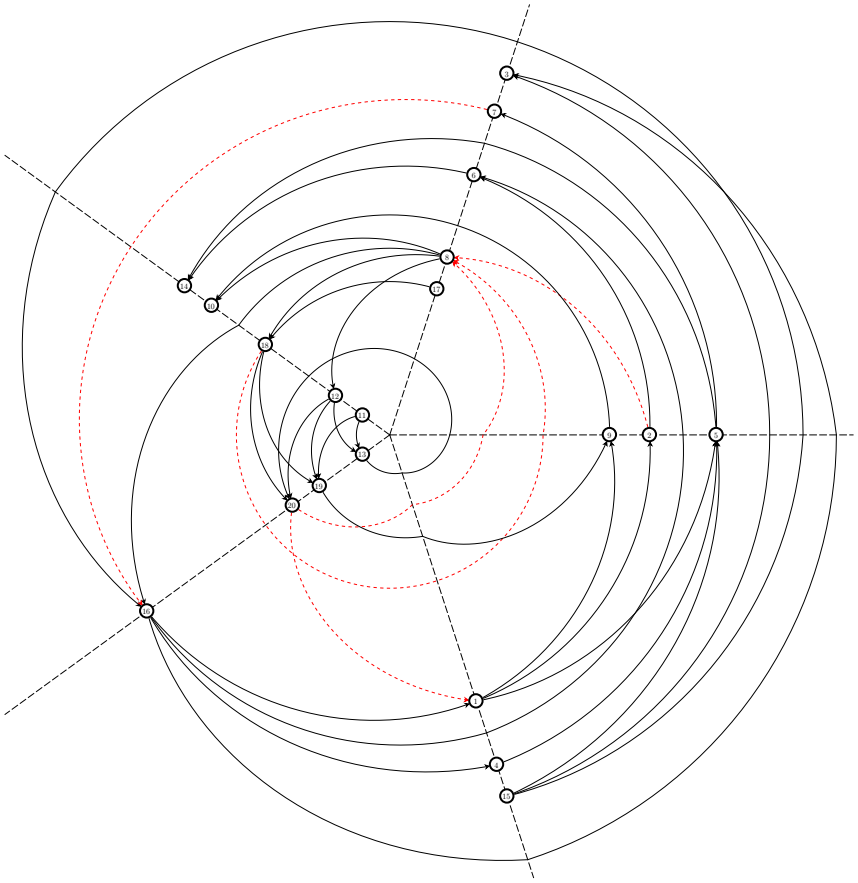


Figure 18: 2D drawing of G_{20} produced by one balanced run

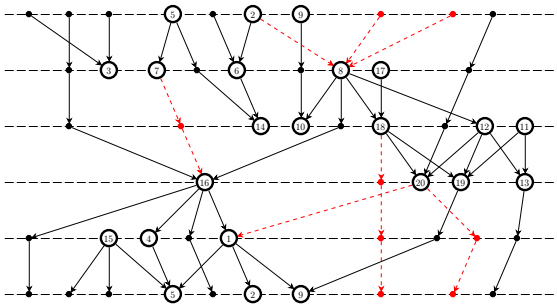


Figure 19: Intermediate drawing, surface of the 3D drawing of G_{20} (one run)

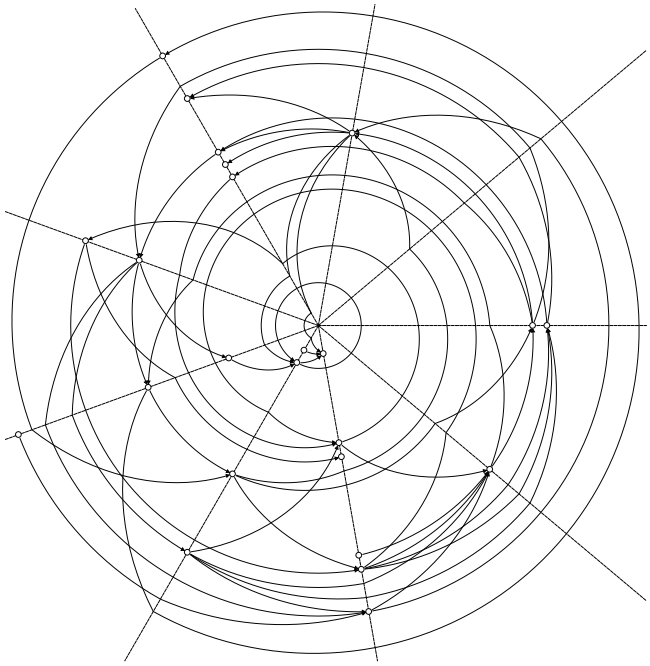


Figure 20: 2D drawing of G_{24} produced by one balanced run

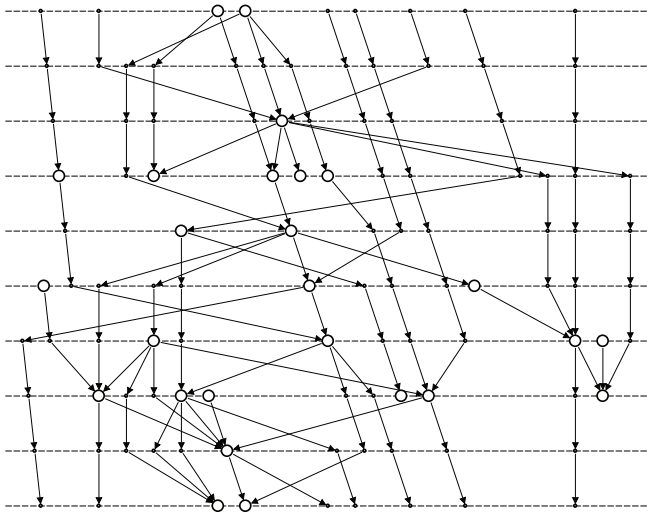
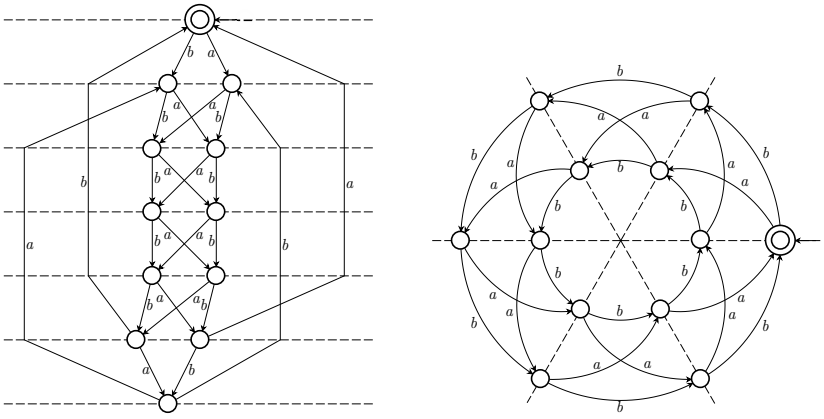


Figure 21: Intermediate drawing and surface of the 3D drawing of G_{24} (one run)



(a) Hierarchical drawing

(b) Cyclic drawing

Figure 22: Drawings of a deterministic finite automaton (DFA) accepting the language $L = \{\, w \in \{a,b\}^* \mid |w| = 0 \pmod 6 \wedge |w|_a = 0 \pmod 2 \,\}$, where $|w|_a$ counts the symbols a within the word w

References

[1] C. Bachmaier. A radial adaption of the Sugiyama framework for visualizing hierarchical information. *IEEE Trans. Vis. Comput. Graphics*, 13(3):583–594, 2007.

[2] C. Bachmaier. *A Generalized Framework for Drawing Directed Graphs.* Habilitation thesis, University of Passau, 2009.

[3] C. Bachmaier, F. J. Brandenburg, W. Brunner, and F. Hübner. Global k-level crossing reduction. *J. Graph Alg. App.*, 2011. To appear.

[4] C. Bachmaier, F. J. Brandenburg, W. Brunner, and G. Lovász. Cyclic leveling of directed graphs. In I. G. Tollis and M. Patrignani, editors, *Proc. Graph Drawing, GD 2008*, volume 5417 of *LNCS*, pages 348–359. Springer, 2009.

[5] C. Bachmaier, F. J. Brandenburg, M. Forster, P. Holleis, and M. Raitner. Gravisto: Graph visualization toolkit. In J. Pach, editor, *Proc. Graph Drawing, GD 2004*, volume 3383 of *LNCS*, pages 502–503. Springer, 2004.

[6] C. Bachmaier and W. Brunner. Linear time planarity testing and embedding of strongly connected cyclic level graphs. In D. Halperin and K. Mehlhorn, editors, *ESA 2008*, volume 5193 of *LNCS*, pages 136–147. Springer, 2008.

[7] J. Bang-Jensen and G. Z. Gutin. *Digraphs: Theory, Algorithms and Applications.* Monographs in Mathematics. Springer, 2007.

[8] 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, 2002.

[9] W. Brunner. *Cyclic Level Drawings of Directed Graphs.* Dissertation, University of Passau, 2010.

[10] C. Buchheim, M. Jünger, and S. Leipert. A fast layout algorithm for k-level graphs. In J. Marks, editor, *Proc. Graph Drawing, GD 2000*, volume 1984 of *LNCS*, pages 229–240. Springer, 2001.

[11] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs.* Prentice Hall, 1999.

[12] P. Eades, X. Lin, and R. Tamassia. An algorithm for drawing hierarchical graphs. *Internat. J. Comput. Geom. Appl.*, 6:145–156, 1996.

[13] P. Eades and K. Sugiyama. How to draw a directed graph. *J. Inform. Process.*, 13(4):424–437, 1990.

[14] P. Eades and N. C. Wormald. Edge crossings in drawings of bipartite graphs. *Algorithmica*, 11(1):379–403, 1994.

[15] M. Eiglsperger, M. Siebenhaller, and M. Kaufmann. An efficient implementation of Sugiyama's algorithm for layered graph drawing. *J. Graph Alg. App.*, 9(3):305–325, 2005.

[16] E. R. Gansner, E. Koutsofios, S. North, and K.-P. Vo. A technique for drawing directed graphs. *IEEE Trans. Software Eng.*, 19(3):214–230, 1993.

[17] E. R. Gansner, S. C. North, and K.-P. Vo. DAG, a program that draws directed graphs. *Software Pract. Exper.*, 17(1):1047–1062, 1988.

[18] M. R. Garey and D. S. Johnson. *A Guide to the Theory of NP-Completeness*. W. H. Freemann, New York, 1979.

[19] M. Jünger, E. K. Lee, P. Mutzel, and T. Odenthal. A polyhedral approach to the multi-layer crossing minimization problem. In G. Di Battista, editor, *Proc. Graph Drawing, GD 1997*, volume 1353 of *LNCS*, pages 13–24. Springer, 1997.

[20] M. Kaufmann and D. Wagner. *Drawing Graphs*, volume 2025 of *LNCS*. Springer, 2001.

[21] K. Mehlhorn and W. Rülling. Compaction on the torus. *IEEE Trans. Comput.-aided Des. Integr. Circuits Syst.*, 9:389–397, 1990.

[22] G. Michal, editor. *Biochemical Pathways: An Atlas of Biochemistry and Molecular Biology*. Wiley, 1998.

[23] C. Pich. Drawing directed graphs clockwise. In D. Eppstein and E. R. Gansner, editors, *Proc. Graph Drawing, GD 2009*, volume 5849 of *LNCS*, pages 369–380. Springer, 2010.

[24] G. Sander. Graph layout through the VCG tool. In R. Tamassia and I. G. Tollis, editors, *Proc. Graph Drawing, GD 1994*, volume 894 of *LNCS*, pages 194–205. Springer, 1995.

[25] G. Sander. A fast heuristic for hierarchical Manhattan Layout. In F. J. Brandenburg, editor, *Proc. Graph Drawing, GD 1995*, volume 1027 of *LNCS*, pages 447–458. Springer, 1996.

[26] G. Sander. Graph layout for applications in compiler construction. *Theor. Comput. Sci.*, 217:175–214, 1999.

[27] P. Serafini and W. Ukovich. A mathematical model for periodic scheduling problems. *SIAM J. Discrete Math.*, 2(4):550–581, 1989.

[28] H. Stamm. On feedback problems in planar digraphs. In *Graph-Theoretic Concepts in Computer Science*, volume 484 of *LNCS*, pages 79–89. Springer, 1991.

[29] K. Sugiyama. *Graph Drawing and Applications for Software and Knowledge Engineers*, volume 11 of *Software Engineering and Knowledge*. World Scientific, 2002.

[30] K. Sugiyama and K. Misue. A simple and unified method for drawing graphs: Magnetic-spring algorithm. In R. Tamassia and I. G. Tollis, editors, *Proc. Graph Drawing, GD 1994*, volume 894 of *LNCS*, pages 364–375. Springer, 1995.

[31] K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierarchical system structures. *IEEE Trans. Syst., Man, Cybern.*, 11(2):109–125, 1981.