
Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 0, no. 0, pp. 0–0 (0)
DOI:

Generalized Layerings for Arbitrary and Fixed Drawing Areas

Ulf Rüegg[1] *Thorsten Ehlers*[1] *Miro Spönemann*[2]
Reinhard von Hanxleden[1]

[1]Dept. of Computer Science, Kiel University, Kiel, Germany
[2]TypeFox GmbH, Kiel, Germany

Abstract

The Directed Layering Problem (DLP) solves a step of the widely used layer-based approach to automatically draw directed acyclic graphs. To cater for cyclic graphs, usually a preprocessing step is used that solves the Feedback Arc Set Problem (FASP) to make the graph acyclic before a layering is determined.

Here we present the Generalized Layering Problem (GLP), which solves the combination of DLP and FASP simultaneously, allowing general graphs as input. We present an integer programming model and a heuristic to solve the NP-complete GLP and perform thorough evaluations on different sets of graphs and with different implementations for the steps of the layer-based approach. We observe that GLP reduces the number of dummy nodes significantly, can produce more compact drawings, and improves on graphs where DLP yields poor aspect ratios.

The drawings resulting from GLP also turn out to be more suitable for making the best possible use of a given drawing area. However, we show that a specialized variant of GLP can yield considerable improvements w. r. t. this particular optimization goal.

Keywords: layer-based layout, Sugiyama layout, layer assignment, integer programming, compactness, aspect ratio

Submitted:	Reviewed:	Revised:	Accepted:	Final:
		Published:		
	Article type:	Communicated by:		

This work was supported by the German Research Foundation under the project *Compact Graph Drawing with Port Constraints* (ComDraPor, DFG HA 4407/8-1).

E-mail addresses: uru@informatik.uni-kiel.de (Ulf Rüegg) the@informatik.uni-kiel.de (Thorsten Ehlers) miro.spoenemann@typefox.io (Miro Spönemann) rvh@informatik.uni-kiel.de (Reinhard von Hanxleden)

1 Introduction

The layer-based approach is a well-established and widely used method to automatically draw directed graphs. It is based on the idea to assign nodes to consecutive *layers* that show the inherent direction of the graph, see Figure 1a for an example. The approach was introduced by Sugiyama et al. [35] and remains a subject of ongoing research.

Given a directed graph, the layer-based approach was originally defined for acyclic graphs as a pipeline of three phases. However, two additional phases are necessary to allow practical usage, which are marked with asterisks:

1. *Cycle removal*:* Eliminate all cycles by reversing a preferably small subset of the graph's edges. This phase adds support for cyclic graphs as input.

2. *Layer assignment:* Assign all nodes to numbered *layers* such that edges point from layers of lower index to layers of higher index. Edges connecting nodes that are not on consecutive layers are split by so-called *dummy nodes*.

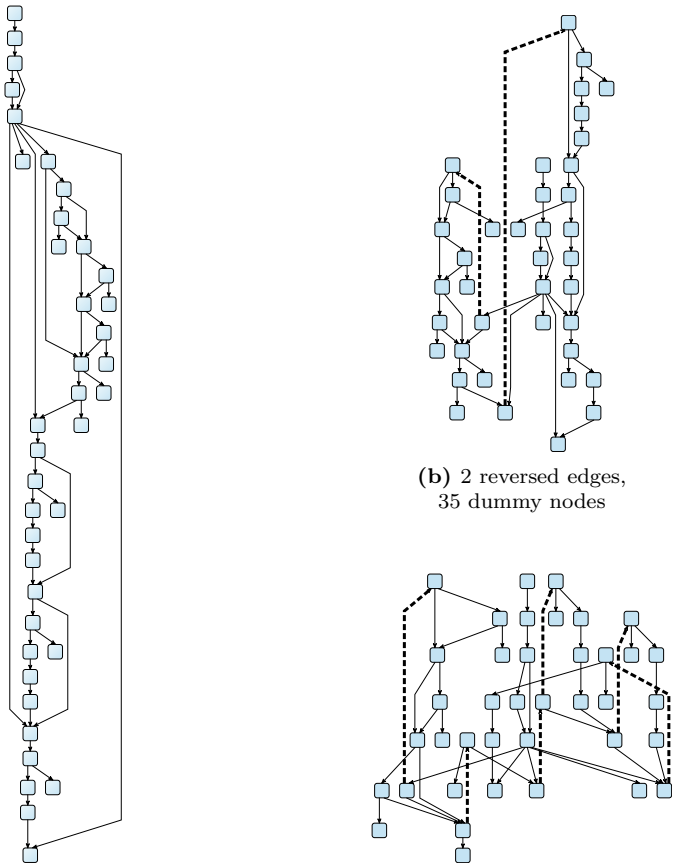
3. *Crossing reduction:* Find an ordering of the nodes within each layer such that the number of edge crossings is minimized.

4. *Coordinate assignment:* Determine explicit node coordinates with the goal to minimize the distance of edge endpoints.

5. *Edge routing*:* Compute bend points for edges, e. g. with an orthogonal style.

While state-of-the-art methods produce drawings that are often satisfying, there are graph instances where the results show bad *compactness* and unfavorable *aspect ratio* [16]. In particular, the number of layers is bound from below by the longest path of the graph that results from the first phase. When placing the layers vertically one above the other, this affects the height of the drawing, see Figure 1a. Also, decisions made during the first phase may turn out to be disadvantageous for the second phase.

The layer-based approach is explicitly designed for directed graphs, emphasizing inherent directionality. Nevertheless, the approach can be used to lay out undirected graphs and to lay out graphs where the direction is of minor or no importance. Drawings of a graph are displayed on some medium, for instance a computer screen, or are printed on paper. One usually desires a drawing that uses the available space to its full potential. For this to happen, the layout algorithm must specifically tailor the drawing for the selected medium. A question immediately following is whether it is required that a user can tell if a sequence of drawings originates from the same graph. Consider Figure 1. Drawing (a) lends itself well to be printed on a paper in portrait orientation, (c) lends itself better for a landscape orientation. However, it is not easy to verify that the different drawings represent the same graph.



(a) 0 reversed edges, 71 dummy nodes

(b) 2 reversed edges, 35 dummy nodes

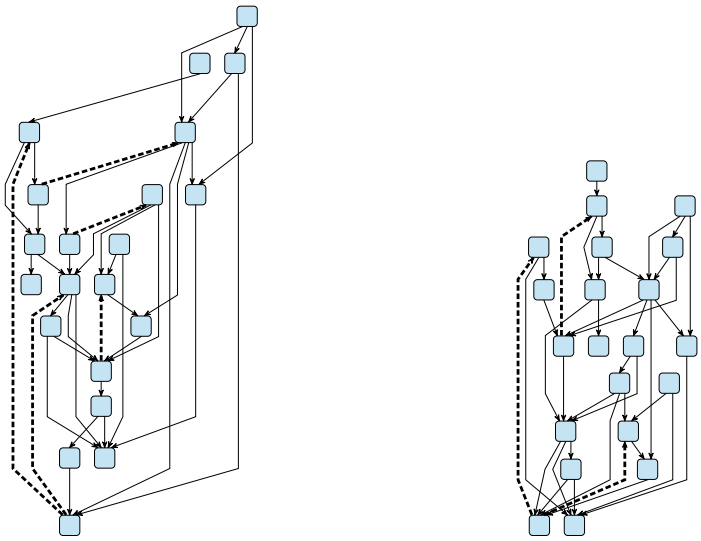
(c) 6 reversed edges, 16 dummy nodes

Figure 1. Different drawings of the g.39.29 graph from the North graphs collection [8]. (a) is drawn with known methods [14], (b) and (c) are results of the methods presented here. Reversed edges are drawn bold and dashed.

We conclude from these observations that there are at least two criteria that must be evaluated in practice when selecting an appropriate layout strategy for the individual use case: (a) the importance of a uniform edge direction, and (b) the recognizability of a graph across different drawings of it. The variety of possible layout objectives arising out of these criteria is too large to address them fully in one paper. We restrict ourselves therefore as explained next.

Contributions. The focus of this paper is on the first two phases described above. They determine the initial topology of the drawing and thus directly impact the compactness and the aspect ratio of the drawing.

We introduce a new layer assignment method that can handle cyclic graphs and is able to consider compactness properties for selecting an edge reversal set.



(a) 5 reversed edges, 55 dummy nodes **(b)** 3 reversed edges, 34 dummy nodes

Figure 2. A graph drawn with (a) EaGa (known methods as described in Section 7) and (b) 1-30-GLP (this work). This example illustrates that our method can perform better in both metrics: reversed edges (dashed and bold) and number of dummy nodes.

Specifically, 1) it can overcome the previously mentioned lower bound on the number of layers arising from the longest path of a graph, 2) it can be flexibly configured to either favor elongated or narrow drawings, thus improving on aspect ratio, and 3) compared to previous methods it is able to reduce both the number of dummy nodes and reversed edges for certain graphs simultaneously. See Figure 1 and Figure 2 for examples.

We discuss how to solve the underlying objective of the new method to optimality using an integer programming model as well as heuristically, and evaluate both. The evaluations serve as a basis to assess the flexibility of our methods to target explicit drawing areas, such as a computer screen. Using a specialized variant of our new method, we find that there is room for improvements w. r. t. this particular optimization goal and give suggestions for future research directions. The latter two points (Section 6) were not yet included in the conference version of this paper [30].

Outline. We introduce problems and definitions in Section 2, and present methods to solve the newly introduced problems in Sections 3 and 4. Sections 5 and 6 discuss thorough evaluations with regards to an arbitrary and fixed drawing area, respectively. Section 7 discusses the relevant literature as well as open questions. We conclude in Section 8.

2 Definitions and Problem Classification

Let $G = (V, E)$ denote a graph with a set of nodes V and a set of edges E. We write an edge between nodes u and v as (u, v) if we care about direction, as $\{u, v\}$ otherwise. A *layering* of a directed graph G is a mapping $L : V \to \mathbb{N}$. A layering L is *valid* if $\forall (u, v) \in E$: $L(v) - L(u) \geq 1$.

Problem 1 (Directed Layering (DLP)) *Let $G = (V, E)$ be an acyclic directed graph. The problem is to find a minimum k and a valid layering L such that $\sum_{(v,w)\in E}(L(w) - L(v)) = k$.*

DLP was originally introduced by Gansner et al. [14]. We extend the idea of a layering for directed acyclic graphs to general graphs, i. e. graphs that are either directed or undirected and that can possibly be cyclic. Undirected graphs can be handled by assigning an arbitrary direction to each edge, thus converting it into a directed one, and by hardly penalizing reversed edges. We call a layering L of a general graph G *feasible* if $\forall \{u, v\} \in E : |L(u) - L(v)| \geq 1$.

Problem 2 (Generalized Layering (GLP)) *Let $G = (V, E)$ be a possibly cyclic directed graph and let $\omega_{\text{len}}, \omega_{\text{rev}} \in \mathbb{N}$ be weighting constants. The problem is to find a minimum k and a feasible layering L such that*

$$\omega_{\text{len}} \left(\sum_{(v,w)\in E} |L(w) - L(v)| \right) + \omega_{\text{rev}} \left| \{(v, w) \in E : L(v) > L(w)\} \right| \;=\; k \;.$$

Intuitively, the left part of the sum represents the overall edge length (i. e. the number of dummy nodes) and the right part represents the number of reversed edges (i. e. the FAS). After reversing all edges in this FAS, the feasible layering becomes a valid layering. Compared to the standard cycle removal phase combined with DLP, the generalized layering problem allows more flexible decisions on which edges to reverse. Also note that GLP with $\omega_{\text{len}} = 1, \omega_{\text{rev}} = \infty$ is equivalent to DLP for acyclic input graphs and that while DLP is solvable in polynomial time, both parts of GLP are NP-complete [29].

3 The IP Approach

In the following, we describe how to solve GLP using integer programming. The rough idea of this model is to assign an integer value to each node of the given graph that represents the layer in which that node is to be placed.

Input and parameters. Let $G = (V, E)$ be a graph with node set $V = \{1, \ldots, n\}$. Let e be the adjacency matrix, i. e. $e(u, v) = 1$ if $(u, v) \in E$ and $e(u, v) = 0$ otherwise. ω_{len} and ω_{rev} are weighting constants.

Integer decision variables. $l(v)$ takes a value in $\{1, \ldots, n\}$ indicating that node v is placed in layer $l(v)$, for all $v \in V$.

Boolean decision variables. $r(u,v) = 1$ if and only if edge $e = (u,v) \in E$ and e is reversed, i. e. $l(u) > l(v)$, for all $u, v \in V$. Otherwise, $r(u,v) = 0$.

$$\text{Minimize} \qquad \omega_{\text{len}} \sum_{(u,v)\in E} |l(u) - l(v)| + \omega_{\text{rev}} \sum_{(u,v)\in E} r(u,v).$$

The sums represent the edge lengths, i. e. the number of dummy nodes, and the number of reversed edges, respectively. Constraints are defined as follows:

$$1 \leq l(v) \leq n \qquad \forall v \in V \tag{A}$$

$$|l(u) - l(v)| \geq 1 \qquad \forall (u,v) \in E \tag{B}$$

$$n \cdot r(u,v) + l(v) \geq l(u) + 1 \qquad \forall (u,v) \in E \tag{C}$$

Constraint (A) restricts the range of possible layers. (B) ensures that the resulting layering is feasible. (C) binds the decision variables in r to the layering, i. e. because r is part of the objective, and $\omega_{\text{rev}} > 0$, $r(u,v)$ gets assigned 0 unless $l(v) < l(u)$, for all $(u,v) \in E$.

Variations. The model can easily be extended to restrict the number of layers by replacing the n in constraint (1) by a desired bound $b \leq n$. The edge matrix can be extended to contain a weight $w_{u,v}$ for each edge $(u,v) \in E$. This can be helpful if further semantic information is available, i. e. about feedback edges that lend themselves well to be reversed.

Jabrayilov et al. present two MIP models for slight variations of GLP [21]. The first one, called CGL, considers the contribution of dummy nodes to a layer's width and adds the width of the layering to the objective function. Consequently, a reasonable bound on the height must be set which is part of the input. The second one, called MML, neglects the contribution of dummy nodes to a layer's width and maximizes the length of reversed edges, significantly improving on execution time. For certain use cases the reversed edge length maximization may be desired.

4 The Heuristic Approach

Interactive modeling tools providing automatic layout facilities require execution times significantly shorter than one second. As the IP formulation discussed in the previous section rarely meets this requirement, we present a heuristic to solve GLP. It proceeds as follows. 1) Leaf nodes are removed iteratively, since it is trivial to place them with minimum edge length and desired edge direction. Note that therefore the heuristic is not able to improve on trees that yield a poor compactness. For reasons we explain in Section 7 we leave this for future research. 2) For the (possibly cyclic) input graph an initial feasible layering is constructed which is used to deduce edge directions yielding an acyclic graph. 3) Using the network simplex method presented by Gansner et al. [14], a solution with minimal edge length is created. 4) We execute a greedy improvement procedure after which we again deduce edge directions and re-attach the leaves.

Algorithm 1. constructLayering

```
Input: directed graph G = (V, E)
Data: Sets U, C. For all v ∈ V score[v], incAs[v], outAs[v]
lIndex ← −1, rIndex ← 0
Output: index[v]: feasible layering of G
1  for v ∈ V do
2      score[v] ← |{w | {v, w} ∈ E}|
3      incAs[v] ← 0
4      outAs[v] ← 0
5      add v to U
6  remove random v from U
7  c ← v
8  add c to C
9  while U not empty do
10     if incAs[c] < outAs[c] then
11         index[c] ← lIndex−−
12     else
13         index[c] ← rIndex++
14     remove c from U and C
15     cScore ← ∞
16     for v ∈ {w | {c, w} ∈ E ∧ w ∈ U} do
17         add v to C
18         score[v]−−
19         if (c, v) ∈ E then incAs[v]++ else outAs[v]++
20     for v ∈ C do
21         if score[v] < cScore then cScore ← score[v]
22         c ← v
```

5) We apply the network simplex algorithm a second time to get a valid layering with minimal edge lengths for the next steps of the layer-based approach. In the following we will discuss steps 2 and 4 in further detail.

Step 2: Layering Construction. To construct an initial feasible solution we follow an idea that was first presented by McAllister as part of a greedy heuristic for the Linear Arrangement Problem (LAP) [23] and later extended by Pantrigo et al. [28].

Nodes are assigned to distinct indexes, where as a start, a node is selected randomly, assigned to the first index, and added to a list of assigned nodes. Based on the list of assigned nodes a candidate list is formed, and the most promising node is assigned to the next index. As decision criterion we use the difference between the number of edges incident to unassigned nodes and the number of edges incident to assigned nodes. This procedure is repeated until all nodes are assigned to distinct indexes (see Algorithm 1).

In contrast to McAllister, for GLP we allow nodes to be added to either side of the list of assigned nodes, and decide the side based on the number of reversed edges that would emerge from placing a certain node on that side. For this we use a decreasing left index variable and an increasing right index variable.

Step 4: Layering Improvement. At this point a feasible layering with a minimum number of dummy nodes w. r. t. the chosen FAS is given since we execute the network simplex method of Gansner et al. beforehand. Thus we can

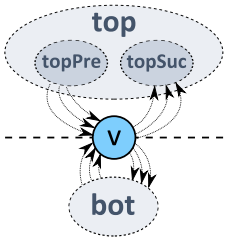


Figure 3. Illustrating the regions used by the heuristic.

only improve on the number of reversed edges. We determine possible *moves* and decide whether to take the move based on a *profit* value. Let a graph $G = (V, E)$ and a feasible layering L be given. For ease of presentation, we define the following notions, which are illustrated in Figure 3. An example: For a node v, $_{topSuc}$ are the nodes connected to v via an outgoing edge of v and are currently assigned to a layer with lower index than v's index. Intuitively, $_{topSuc}$ (just as $_{botPre}$) are nodes connected by an edge pointing into the "wrong" direction.

$$v.topSuc = \{w : (v,w) \in E \wedge L(v) > L(w)\} \quad v.botSuc = \{w : (v,w) \in E \wedge L(v) < L(w)\}$$
$$v.topPre = \{w : (w,v) \in E \wedge L(w) < L(v)\} \quad v.botPre = \{w : (w,v) \in E \wedge L(w) > L(v)\}$$
$$v.topAdj = v.topSuc \cup v.topPre \qquad v.botAdj = v.botSuc \cup v.botPre$$

For all these functions we define suffixes that allow to query for a certain set of nodes before or after a certain index. For instance, for all top successors of v before index i we write $v._{topSucBefore(i)} = \{w : w \in v.topSuc \wedge L(w) < i\}$.

Let $move : V \to \mathbb{N}$ denote a function assigning to each node a natural value. The function describes whether it is possible to move a node without violating the layering's feasibility as well as how far the node should be moved. For instance, let $_{topPre}$ be empty for a node v, and $_{topSuc}$ be non-empty. Thus, we can move v to an arbitrary layer with lower index than $L(v)$. A good choice would be one layer before any $w \in v._{topSuc}$ since this would alter the incident edges to point downwards.

$$move(v) = \begin{cases} 0 & \text{if } v.topSuc = \emptyset, \\ L(v) - \min(\{L(w) : w \in v.topSuc\}) + 1 & \text{if } v.topPre = \emptyset, \\ L(v) - \max(\{L(w) : w \in v.topPre \setminus v.topSuc\}) - 1 & \text{otherwise,}[1] \end{cases}$$

where $\max(\emptyset) = L(v) - 1$.

Let $profit : V \times \mathbb{N} \times \mathbb{N} \to \mathbb{Z}$ denote a function assigning a quality score to a node v that decides if it is worth to move v and increase the length of some edges for a subset of the edges to point downwards. For this the natural m is computed by the *move* function defined above and x represents the layer v would be moved to. Note that we reuse ω_{len} and ω_{rev} here but do not expect

[1] During the evaluations $L(v) - \max(\{L(w) : w \in v.topPre\}) - 1$ was used as third case. This did not necessarily preserve the layering's feasibility but was turned into a valid layering by the subsequent network simplex execution in either case.

Algorithm 2. improveLayering

```
   Input: feasible layering of G = (V, E) in index[v]
   Data: priority queue PQ
   For all v ∈ V move[v], profit[v]
   Output: index[v]: feasible layering of G
 1 for v ∈ V do
 2 |   move[v] ← move(v)
 3 |   profit[v] ← profit(v, move[v], index[v] − move[v])
 4 |   if profit[v] > 0 then enqueue v to PQ
 5 while PQ not empty do
 6 |   v ← dequeue PQ
 7 |   index[v] −= move[v]
 8 |   for w ∈ {w | {v, w} ∈ E} do
 9 |   |   update move[w] and profit[w]
10 |   |   if profit[w] > 0 then
11 |   |   |   enqueue/update w to/in PQ
12 |   |   else
13 |   |   |   possibly dequeue w from PQ
```

them to have an impact as strong as for the IP. For the rest of the paper we fix them to 1 and 5.

$$\mathit{profit}(v,m,x) = \begin{cases} 0 & \text{if } m \leq 1, \\ \omega_{\text{len}}(m|v.\mathit{topAdjBefore(x)}| - m|v.\mathit{botAdj}|) & \\ \quad + \ \omega_{\text{rev}}|v.\mathit{topSucAfter(x)}| & \text{otherwise.} \end{cases}$$

As seen in Algorithm 2, the *move* and *profit* functions are determined initially for a given feasible layering. A queue, sorted based on profit values, is then used to successively perform moves that yield a profit. After a move of node n, both functions can be updated for all nodes in the adjacency of n.

Time Complexity. Removing leaf nodes requires linear time, $O(|V| + |E|)$. Algorithm 1 is quadratic in the number of nodes, $O(|V|^2)$. The while loop has to assign an index to every node and the two inner for loops are, for a complete graph, iterated $\frac{|V|}{2}$ times on average. Determining the next candidate (lines 15–16) could be accelerated using dedicated data structures. The improvement step strongly depends on the input graph. The network simplex method runs reportedly fast in practice [14], although it has not been proven to be polynomial. Our evaluations showed that the heuristic's overall execution time is clearly dominated by the network simplex method (cf. Section 5).

5 Evaluation

In this section we evaluate three points: 1) the general feasibility of GLP to improve the compactness of drawings, 2) the quality of metric estimations for area and aspect ratio, and 3) the performance of the presented IP and heuristic. Our main metrics of interest here are height, area, and aspect ratio, as defined in an earlier paper [16]. Remember that the layer-based approach is defined as a pipeline of several independent steps. Regarding the result of the layering

phase, which is the focus of our research here, area and aspect ratio can only be estimated using the number of dummy nodes, the number of layers, and the maximal number of nodes in a layer. Results can be seen in Table 1 and Table 2, which we will discuss in more detail in the remainder of this section.

Obtaining a Final Drawing. In order to collect all metrics we desire, we have to create a final drawing of a graph. Over time numerous strategies have been presented for each step of the layer-based approach, we thus present several alternatives. To break cycles we use a popular heuristic by Eades et al. [11]. To determine a layering we use our newly presented approach GLP (both the IP method and heuristic, denoted by GLP-IP and GLP-H) and alternatively the network simplex method presented by Gansner et al. [14]. We denote the combination of the cycle removal of Eades et al. and the layering of Gansner et al. as EaGa and consider it to be an alternative to GLP. Crossings between pairs of layers are minimized using a layer sweep method in conjunction with the barycenter heuristic, as originally proposed by Sugiyama et al. [35]. We employ two different strategies to determine fixed coordinates for nodes within the layers. First, we consider a method introduced by Buchheim et al. that was extended by Brandes and Köpf [4, 2], which we denote as BK. Second, we use a method inspired by Sander [32] that we call LS. Edges are routed either using polylines (Poly) or orthogonal segments (Orth). The orthogonal router is based on the methods presented by Sander [33]. Overall, this gives twelve setups of the algorithm: three layering methods, two coordinate assignment algorithms, and two edge routing procedures. In the following, let ω_{len}-ω_{rev}-GLP denote the used weights. If we do not further qualify GLP, we refer to the IP model.

Test Graphs. Our new approach is intended to improve the drawings of graphs with a large height and relatively small width, hence unfavorable aspect ratio. Nevertheless, we also evaluate the generality of the approach using a set of 160 randomly generated graphs with 17 to 60 nodes and an average of 1.5 edges per node. The graphs were generated by creating a number of nodes, assigning out-degrees to each node such that the sum of outgoing edges is 1.5 times the nodes, and finally creating the outgoing edges with a randomly chosen target node. Unconnected nodes were removed. Second, we filtered the graph set provided by North[2] [8] based on the aspect ratio and selected 146 graphs that have at least 20 nodes and a drawing (created using BK and Poly) with an aspect ratio below 0.5, i. e. are at least twice as high as wide. We also removed plain paths, that is, pairs of nodes connected by exactly one edge, and trees. For these special cases GLP in its current form would not change the resulting number of reversed edges as all edges can be drawn with length 1. This is also true for any bipartite graph. Note however that GLP can easily incorporate a bound on the number of layers which can straightforwardly be used to force more edges to be reversed, resulting in a drawing with better aspect ratio.

[2] `http://www.graphdrawing.org/data/`

Table 1. Average values for different layering strategies employed to the test graphs. Different weights are used for GLP-IP as specified in the column head and final drawings were created using BK and Poly. For GLP-H* no improvement was performed. A detailed version of these results is included in the appendix (cf. A.2).

(a) Random graphs

	1-10	1-20	1-30	1-40	1-50	EaGa	GLP-H	GLP-H*
Reversed edges	3.71	2.89	2.64	2.54	2.44	2.93	8.67	10.36
Dummy nodes	34.45	46.73	52.79	56.14	60.53	72.64	48.48	58.21
Edge crossings	2.41	2.18	2.10	2.04	2.02	1.98	1.93	1.84
Height	843	943	980	1,004	1,025	1,084	930	1,027
Area	631,737	672,717	691,216	700,385	708,361	737,159	656,070	720,798
Aspect ratio	0.77	0.65	0.63	0.61	0.59	0.55	0.67	0.60

(b) North graphs

	1-10	1-20	1-30	1-40	1-50	EaGa	GLP-H	GLP-H*
Reversed edges	2.74	1.47	1.02	0.72	0.56	0	7.07	8.55
Dummy nodes	39.91	55.47	65.73	75.66	82.47	141.30	53.53	68.91
Edge crossings	1.46	1.23	1.13	1.06	0.99	0.86	1.19	1.12
Height	1,068	1,224	1,334	1,409	1,469	1,727	1,137	1,216
Area	587,727	622,838	641,581	660,842	695,494	874,374	629,778	691,372
Aspect ratio	0.34	0.28	0.24	0.23	0.22	0.20	0.33	0.32

General Feasibility of GLP. An exemplary result of the GLP approach compared to EaGa can be seen in Figure 2. For that specific drawing, GLP produces fewer reversed edges, fewer dummy nodes, and less area (both in width and height). For all tested setups the average effective height and area (normalized by the number of nodes) of GLP and the heuristic are smaller than EaGa's, see Table 2. The average aspect ratios come closer to 1.0. For simplicity, in this paper we desire aspect ratios closer to 1.0. For a more detailed discussion on this topic see Gutwenger et al. [16].

Furthermore, we found that by altering the weights ω_{rev} and ω_{len} a trade-off between reversed edges and resulting dummy nodes (and thus area and aspect ratio) can be achieved, which can be seen in Table 1a for the random graphs.

The results for the North graphs are similar. Since the North graphs are acyclic, the cycle removal phase is not required and current layering algorithms cannot improve the height. The GLP approach, however, can freely reverse edges and hereby change the height and aspect ratio. Results can be seen in Table 1b. Clearly, EaGa has no reversed edges as all graphs are acyclic. 1-10-GLP starts with an average of 2.7 reversed edges and the value constantly decreases with an increased weight on reversed edges. The number of dummy nodes on the other hand constantly decreases from 141.3 for EaGa to 39.9 for 1-10-GLP. For both sets of graphs the average number of edge crossings per edge slightly increases when a lower weight is set for ω_{rev}. This makes sense since more edges have to be routed between a lower number of layers. The average height and average area of the final drawings decrease with an increasing number of reversed edges. For 1-10-GLP the average height and area are 38.2 %

Table 2. Results for final drawings of the set of random graphs, when applying different layout strategies. For GLP-IP $\omega_{\text{len}} = 1$ and $\omega_{\text{rev}} = 30$ were used. Area is normalized by a graph's node count. The most interesting comparisons are between columns where EaGa and GLP use the same strategies for the remaining steps. Detailed results can be found in the appendix (cf. A.2).

Edge routing	Poly						Orth					
Node coord.	BK			LS			BK			LS		
Layering	EaGa	GLP-IP	GLP-H	EaGa	GLP-IP	GLP-H	EaGa	GLP-IP	GLP-H	EaGa	GLP-IP	GLP-H
Height	1,165	1,043	898	943	824	732	790	711	652	817	746	678
Area	20,194	18,683	15,575	12,383	11,035	10,075	13,582	12,642	11,272	10,666	9,917	9,295
Aspect ratio	0.59	0.67	0.67	0.55	0.64	0.64	0.84	0.96	0.90	0.63	0.70	0.68

and 33.8 % smaller than EaGa. The aspect ratio changes from an average of 0.20 for EaGa to 0.34 for 1-10-GLP.

The results show that for the selected graphs, for which current methods cannot improve on height, the weights of the new approach allow to find a satisfying trade-off between reversed edges and dummy nodes. Furthermore, the improvements in compactness stem solely from the selection of weights, not from an upper bound on the number of layers. Naturally, such a bound can further improve the aspect ratio and height.

Metric Estimations. Table 2 presents results that were measured on the final drawing of a graph. As mentioned earlier, these values are not available when analyzing the result of the layering step and estimations are commonly used to deduce the quality of a result. That is, the height is estimated using the number of layers and the width is estimated using the maximum number of nodes in any layer. For our example graphs, the estimated area reduced from 222.9 (EaGa) to 187.4 (1-30-GLP) on average. The estimated aspect ratios increase on average from 0.74 to 0.84. Both tendencies conform to the averaged effective values in Table 2, i. e. GLP-IP and the GLP-H perform better. However, we observed that for 64 % of the graphs the tendency of the estimated area contradicts the tendency of the effective area (using BK and Poly), and 54 % when not considering dummy nodes. That is, for a specific graph the estimated area might be decreased for GLP compared to EaGa but the effective area is increased for GLP (or vice versa). This clearly indicates that an estimation can be misleading. Besides, coordinate assignment and edge routing can have a non-negligible impact on the aspect ratio and compactness of the final drawing.

Performance of the Heuristic. Results for final drawings using the presented heuristic are included in Table 2 and are comparable to 1-30-GLP, i. e. the heuristic performs better than EaGa w. r. t. the desired metrics. Table 1a and Table 1b underline this result and show that the improvement step of the heuristic clearly improves on all measured metrics. More detailed results can be found in the appendix. Nevertheless, the heuristic yields significantly more reversed edges. When aiming for compactness, we consider this to be acceptable.

Execution Times. To solve the IP model we used CPLEX 12.6 and executed the evaluations on a server with an Intel Xeon E5540 CPU and 24 GB memory. The execution times for GLP-IP vary between 476ms for a graph with 19 nodes and 541s for a graph with 58 nodes and exponentially increase with the graph's node count. This is impracticable for interactive from practice that rely on responsive automatic layout, but is fast enough to collect optimal results of medium sized graphs for research purposes.

The execution time of the heuristic is compared to EaGa and was measured on a laptop with an Intel i7-3537U CPU and 8 GB memory. The reported time includes only the first two steps of the layer-based approach. It turns out that the execution time of the heuristic is on average 2.3 times longer than EaGa. This seems reasonable, as it involves two executions of the network simplex layering method. For the tested graphs, the construction and improvement steps of the heuristic hardly contribute to its overall execution time. The effective execution time ranges between 0.1ms and 10.0ms for EaGa and 0.3ms and 19.7ms for the heuristic. The heuristic is fast enough to be used in interactive tools.

We also ran the algorithm five times for five randomly generated graphs with 1000 nodes and 1500 edges. EaGa required an average of 374ms, the heuristic 666ms with about 4ms for construction and 2ms for improvement. This shows that the time contribution of the latter two is negligible even for larger graphs.

6 Targeting a Specific Drawing Area

The previous section showed that the GLP allows to reduce the number of dummy nodes by increasing the number of reversed edges. Adjusting the weights ω_{rev} and ω_{len}, the aspect ratio of the resulting drawing can be influenced to a certain extent. This section addresses the question whether GLP is able to create drawings that are particularly suited for a certain drawing area, e. g. a specific computer screen. The measures used before are not sufficient for this. When seen in isolation, none of the metrics height, width, and area gives any insight on how good a drawing leverages the available drawing area. A perfectly matching aspect ratio can be achieved by enlarging one of the two dimensions of the drawing, for instance by simply increasing the spacing between the elements in that dimension. However, this introduces unnecessary whitespace and does not improve readability.

We propose the following solution. For a given drawing area, we say the "best" drawing is the drawing which can be displayed within the given drawing area with maximum scaling factor. In other words, the elements of the drawing, which may include labels that must be legible by users, should be displayed as large as possible.[3] We call this *max scale measure* and define it as follows:

Let $\mathcal{R} = (r_w, r_h)$ denote the width and height of a reference drawing area and $a_{\mathcal{R}} = \frac{r_w}{r_h}$ its aspect ratio. For instance, an A4 paper sheet (portrait) has $\mathcal{R}_{A4} = (210\text{mm}, 297\text{mm})$ and $a_{\mathcal{R}_{A4}} = 1/\sqrt{2}$. When only the relation of the two

[3] Informally, we called this the "pencil metric" because we used to compare the sizes of labels and nodes on a screen with pencil and thumb.

dimensions to each other is of interest, $\mathcal{R}_{A4}$ can be simplified to $(1, \sqrt{2})$. The *max scale value* s of a drawing D with width w and height h in relation to a reference frame $\mathcal{R}$ then is:

$$s = \min\left\{\frac{r_w}{w}, \frac{r_h}{h}\right\}.$$

Where convenient, e. g. when used as part of a minimization problem, it can alternatively be defined as the inverse of s: $q = \max\left\{\frac{w}{r_w}, \frac{h}{r_h}\right\}$. We use this in the next section.

Additionally, given two drawings D and D' with their max scale values s and s', the *max scale ratio* $r = \frac{s}{s'}$ of the two drawings indicates which of the two drawings can be displayed with a larger scale factor within the given frame. In other words, if $r > 1$, the drawing D can be displayed larger than D'. Thus, from this measure's perspective, D is preferable. To illustrate, a max scale ratio of 2 indicates that the elements of the better drawing can be displayed twice as large as the elements of the inferior one.

Max Scale Values of GLP. We calculated the max scale values of the drawings created for the North graphs as discussed in Section 5 for four different reference frames with aspect ratios of 0.25, 0.5, 1, and 2. The results can be seen in Figure 4 in the form of boxplots; the horizontal bar within a box denotes the median, the dot within a box denotes the average. Note that no comparison can be made across target aspect ratios but only between the results of different methods for the same aspect ratio.

For a target aspect ratio of 0.25 EaGa shows the best max scale values, which is not surprising, since the set of test graphs was selected based on EaGa producing a drawing with an aspect ratio below 0.5. Also, the GLP configurations aim at producing wider drawings. For aspect ratios larger than 0.25, all variants of GLP produce better max scale values than EaGa. Also, the selected weights gradually change the max scale value, which is coherent with the observations for height, area, and aspect ratio in Section 5. However, the configuration with the smallest weight on reversed edges, 1-10-GLP, gives the best max scale values for almost every reference frame. While this is not immediately surprising since 1-10-GLP's drawings are the ones with the smallest area, it prevents a static mapping of weights to reference frames. Additionally, it leads to the question whether significantly larger max scale values are possible, and if so, what trade-offs have to be made for these. We seek to answer the following questions in the remainder of this section:

1. How do GLP's drawings relate to drawings with optimal max scale values? How much better are the optimal ones?

2. How many edges have to be reversed to achieve optimal max scale values?

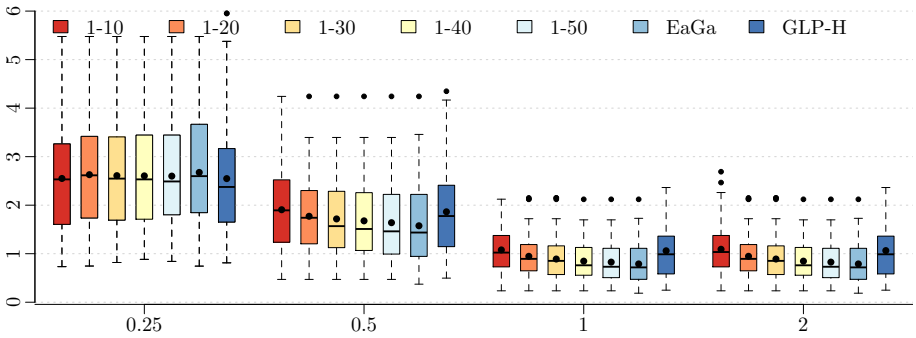


Figure 4. Computed max scale values (y axis) for four different target aspect ratios (x axis), using the results for the tested North graphs and the GLP variants as discussed in Section 5.

6.1 GLP-MS

To answer the previously stated questions we extend GLP and incorporate the max scale measure into its objective:

$$\begin{aligned} \text{Minimize} \quad & \omega_{\text{len}} \sum_{(v,w)\in E} |L(w) - L(v)| \\ & + \omega_{\text{rev}} \ |\{(v,w) \in E : L(v) > L(w)\}| \\ & + \omega_{\text{ms}} \ \max\left\{ \frac{w(L)}{r_w}, \frac{h(L)}{r_h} \right\}, \end{aligned}$$

where $\mathcal{R} = (r_w, r_h)$ denotes the reference frame, $w(L) = \max_{\ell \in L} \ \sum_{v \in \ell} \ w(v)$ the estimated width of the layering, and $h(L) = \sum_{\ell \in L} \ \max_{v \in \ell} \ h(v)$ the estimated height of the layering. $w(v)$ and $h(v)$ denote the width and height of node v. For the evaluations hereafter both are 1.

To solve this modified version, which we call GLP-MS, the IP formulation presented in Section 3 is not particularly suited. The max scale measure requires reasonable estimations of the width and height of the drawing. This is only possible if the contributions of dummy nodes to the layer widths are considered. Thus, as a basis we use a more sophisticated MIP model (CGL) for a variation of GLP as presented by Jabrayilov et al. [21]. The difference is that they set a fixed bound H on the height and add the overall width W of the layering to the objective. The contribution of dummy nodes is considered. They make use of three types of binary variables. First, $y_{v,k}$ equals 1 if and only if $L(v) < k$ for all $v \in V$ and $1 \leq k \leq H$. The reverse variables $y_{k,v}$ are part of the model for ease of understanding but can be eliminated when implementing the model. Second, $r_{u,v}$ equals 1 if and only if the edge (u, v) is reversed. Third, $z_{uv,k}$ equals 1 if and only if the edge (u, v) produces a dummy node in layer k for all $2 \leq k \leq H - 1$.

To use CGL for GLP-MS, we set H to $|V|$, thus removing H from the input, but introduce two new input variables to denote the reference frame, r_w and r_h. Following the notation of Jabrayilov et al., we replace the width term of the objective, $\omega_{wid} \cdot W$, by $\omega_{\text{ms}} \cdot MS$, where MS represents the inverse max scale value to be minimized, and ω_{wid} and ω_{ms} are weighting constants. Further we remove the constraints that are connected to W, but extend the model to properly incorporate MS. Two dummy nodes s and e are added to the graph's nodes, which are used to mark the very first layer and the very last layer. A dummy edge (s, e) is added to the graph's edges. With this, the following constraints are added to bound MS (the full model can be found in Appendix A.3):

$$y_{k,s} = 0 \qquad \text{for all } 1 \leq k \leq H \tag{D}$$

$$y_{k,v} \leq y_{k,e} \qquad \text{for all } 1 \leq k < H, v \in V \setminus \{e\} \tag{E}$$

$$\frac{w(k) - 1}{r_w} \leq MS \qquad \text{for all } 1 \leq k \leq H \tag{F}$$

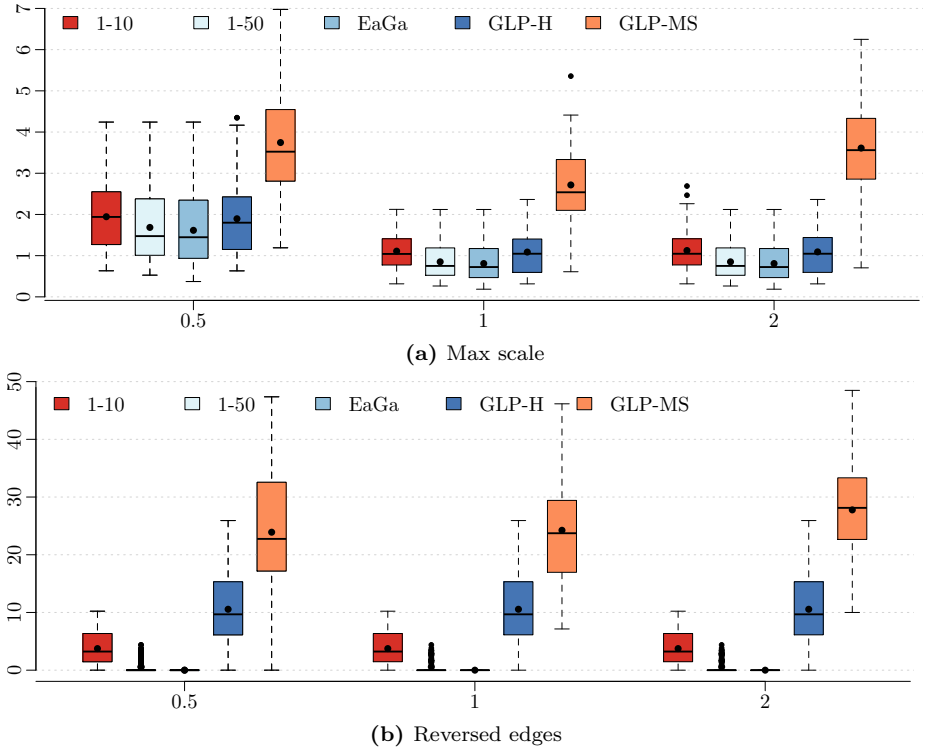
$$\frac{1 + \sum_{1 \leq k \leq H} y_{k,e}}{r_h} \leq MS, \tag{G}$$

where $w(k)$ is the width of layer k and can be computed as shown in Appendix A.3.

Constraints (D) and (E) assure that no original node is placed before and after the s node and e node, respectively. Consequently a dummy node is introduced in every intermediate layer. Since the number of dummy nodes is subject to minimization, s is placed in the first layer and e is placed in the last layer. Constraints (F) restrict MS by the width of the layers. The dummy nodes originating from (s, e) are discarded by the -1. Finally, (G) restricts MS by the layering's height, i. e. it counts the number of layers that are smaller than e's layer and adds the layer itself. The overall width and height of the layering can be modeled as above, since we assume every node and every dummy node to be of unit size.

Results. We executed the model using Gurobi 7[4] on a server with an Intel Xeon E5540 CPU and 24 GB memory. Within the set time limit of 30 minutes, 354 of 438 executions of the North graphs finished. That is, for each of the 146 graphs we ran the model for three different aspect ratios. For 112 of the graphs all three executions finished. Divided by target aspect ratio 2.0, 1.0, and 0.5, executing the model took on average 710, 1.938, and 4.926 seconds, respectively. The results discussed in the following are for the subset of the 112 graphs. The nodes of the North graphs have no specified width and height. We set both to 20 when measuring pixels and assume a unit width and height for GLP-MS. A dummy node contributes the same width to a layer as a regular node.

[4] `http://www.gurobi.com/`



(a) Max scale

(b) Reversed edges

Figure 5. Results of applying GLP-MS with three different target aspect ratios (x axis) to 112 of the North graphs. The number of reversed edges in (b) is in percent.

The weights ω_{len}, ω_{rev}, and ω_{ms} are selected such that the max scale measure is prioritized over both other terms and a reversed edge is penalized stronger than long edges:

$$\omega_{\text{len}} = 1, \quad \omega_{\text{rev}} = \omega_{\text{len}} \ |E| \ |V|, \text{ and } \quad \omega_{\text{ms}} = \omega_{\text{rev}}(1 + |E|) \max\{r_w, r_h\}.$$

Note that care has to be taken when using large weights like this with MIP solvers. A solution is often considered optimal by a solver if the relative gap between the lower and upper bound on the objective is lower than a specified parameter. Therefore, since the objective value is dominated by the max scale part, it can happen that the term measuring the edge length is neglected if the parameter value is too large. Gurobi's default 10^{-4} may cause issues with the graphs we tested, which is why we set it to 10^{-10}.

Remember that the results presented in Figure 4 suggest that while the differently weighted GLP is able to improve the max scale values when aiming for a certain drawing area, no significant improvements can be made when only few edges are reversed. This feels natural, as by reversing a small number of edges the overall character of the graph is not changed. A very "long" graph will never be suited for the opposite kind of drawing area.

The results of applying GLP-MS to the North graphs are shown in Figure 5. They clearly indicate that it is possible to achieve significantly larger max scale values for notably different drawing areas if enough edges may be reversed. For all three tested target aspect ratios 0.5, 1.0, and 2.0 the max scale values created with GLP-MS are on average at least two times larger. Answering Question (1) (p. 13) this indicates that GLP's results are still improvable when it comes to a specific drawing area. Nevertheless, answering Question (2), it must be said that GLP-MS reverses about a quarter of the edges on average. Interestingly, there are cases where fewer edges were reversed the wider the drawings got, i. e. the more the drawing deviated from the original.

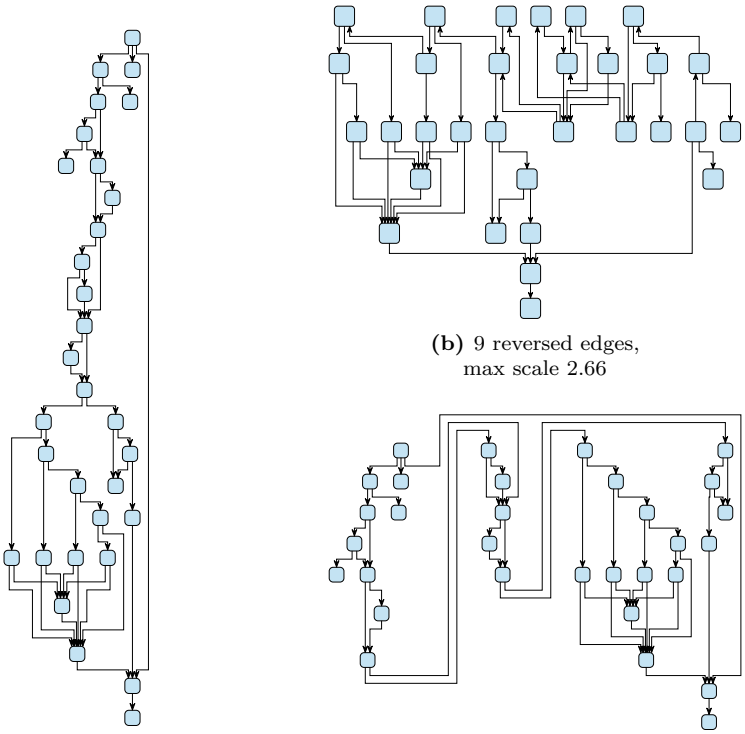
We conclude that for use cases where the directionality is not that important, the compactness of drawings can be improved even further than what GLP achieves. Apart from that we suggest to use a different approach for use cases where directionality should be preserved as much as possible. A higher number of reversed edges seems to be inevitable. They should, however, be clearly distinguishable from the regular edges and therefore be combined at specific areas of the drawing. Consider for example Figure 6. While the traditional drawing (a) is unsuited for a wide screen, drawing (c) better matches the screen's aspect ratio and the well-defined sections where edge point upwards allow to follow the overall flow of the graph easily. This is not true for drawing (b) on the other hand, which nevertheless is the most compact one. We plan to investigate this approach in further detail.

Furthermore, we feel it is important that a method does not depend on input parameters that have to be chosen by users. Weighting factors are only acceptable as inputs to a method as long as they work for the majority of use cases.

7 Related Work and Discussion

The underlying idea of this paper is to combine the initial two phases of the layer-based approach. As explained in the introduction, the cycle removal phase is not part of the original pipeline as suggested by Sugiyama et al. [35] and is a means to an end to handle cyclic graphs. The layering phase is an integral part of the pipeline and significantly affects the appearance of the final drawing. Before we shift our focus to existing layering methods, some words on cycle removal.

The cycle removal phase targets the NP-complete Feedback Arc Set Problem (FASP). In the context of layered graph drawing, several approaches have been proposed to solve FASP either to optimality or heuristically [19]. The most popular heuristic was introduced by Eades et al. [11]. It has been noted before that reversing a minimal number of edges does not necessarily yield the desired results, and application-inherent information might make certain edges better candidates to be reversed [14]. Moreover, the decision which edges to reverse in order to make a graph acyclic has a big impact on the results of the subsequent layering phase. Nevertheless the two phases are executed separately until today.



(a) 0 reversed edges, max scale 1.23

(b) 9 reversed edges, max scale 2.66

(c) 5 reversed edges, max scale 2.4

Figure 6. Different drawings of the g.31.37 graph from the North graphs collection [8]. (a) is drawn with EaGa, (b) with 2-3-GLP. From the three drawings (b) it is the most compact one, however it is not evident that it is the same graph as (a). (c) is created by bluntly cutting the drawing (a) at two points, and placing the chunks next to each other. The stated max scale values are for $\mathcal{R} = (1920, 1080)$, an up-to-date resolution of a wide screen.

Outside the area of graph drawing FASP has been studied extensively and remains a subject for ongoing research, often concerning approximation algorithms and slight variations of the original problem [1, 13, 7, 36, 20]. To give an example, a minimum FAS in a *tournament graph* can help to identify a "fair" ranking of the tournament's participants [5]. In our case, FASP is only one part of the objective function, which implicitly tailors the FAS to allow low numbers of dummy nodes. We therefore used an integer programming approach instead of algorithms specifically designed for FASP as this allows to flexibly combine different optimization criteria. This is particularly relevant as optimum solutions to GLP often contain suboptimal solutions for the underlying FASP. Nevertheless, it would be interesting to see if the desire to have short edges can be integrated into existing FASP algorithms, and to see how results would compare.

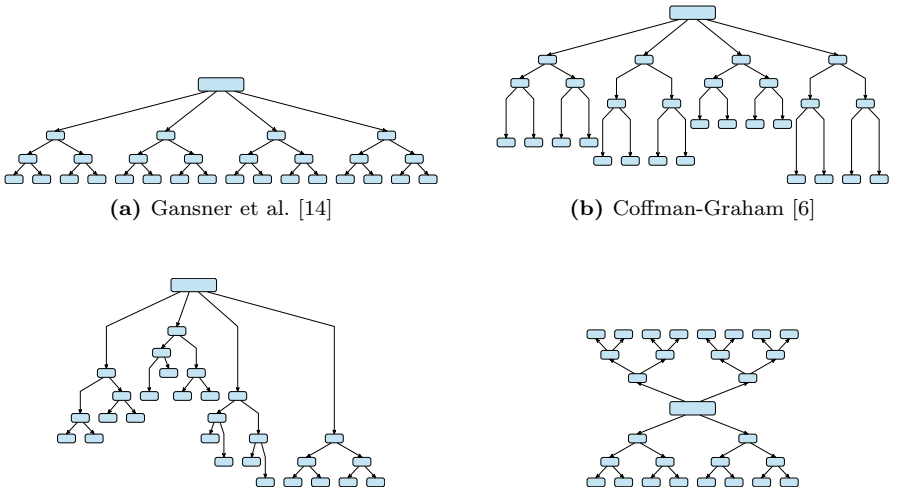
To solve the second phase, i. e. the layer assignment problem, several approaches with different optimization goals have emerged over time.

Traditional. Eades and Sugiyama employ a method that is known as *longest path layering*. It requires linear time and the resulting number of layers equals the number of nodes of the graph's longest path [12]. Gansner et al. solve the layering phase by minimizing the sum of the edge lengths [14]. They show that the problem is solvable in polynomial time and present a network simplex algorithm which in turn is not proven to be polynomial, although it runs fast in practice. Alternatively, their approach can be formulated and solved as the dual of a minimum cost flow problem. This approach was found to inherently produce compact drawings and performed best for the general case in comparison to other layering approaches [18].

Restricted Width. The width of a layering is the maximum number of nodes in any layer. Coffman and Graham's scheduling algorithm with precedence constraints can be used to find a layering with a prescribed bound on the number of original nodes per layer [6]. The algorithm cannot consider the contribution of dummy nodes to a layer's width. Healy and Nikolov tackle the problem of finding a layering subject to bounds on the number of layers and the maximum number of nodes in any layer with consideration of dummy nodes using an integer linear programming approach [18]. The problem is NP-hard, even without considering dummy nodes. In a subsequent paper they present a branch-and-cut algorithm to solve the problem faster and for larger graph instances [17]. Later, Nikolov et al. propose and evaluate several heuristics to find a layering with smaller number of nodes in each layer [26].

Aspect Ratio. Nachmanson et al. present an iterative algorithm to produce drawings with an aspect ratio close to a previously specified value [25]. The idea is to execute all phases of the layer-based approach for a given graph, measure the aspect ratio of the final drawing, and start over if the measured aspect ratio is too far from the desired aspect ratio. Internally they use a slight variation of the Coffman-Graham algorithm to compute a layering, where the input parameter is a real number w. The sum of node widths of any layer must not exceed w. The authors use a binary search to find the best w for the desired aspect ratio. Consequently, their algorithm is not able to advance towards the desired aspect ratio if the unconstrained Coffman-Graham ($w = \infty$) is closest of all possible input parameters. The new methods of this paper can improve the layout in this case. Additionally, here we are interested in a self-contained layering method that does not require an iterative execution of the whole layer-based layout pipeline.

Restricted Height. All of the previously mentioned layering methods have two major drawbacks. 1) They require the input graph to be acyclic upfront, and 2) they are bound to a minimum number of layers equal to the longest path



(a) Gansner et al. [14]

(b) Coffman-Graham [6]

(c) Nikolov et al.; MinWidth heuristic [26]

(d) Alternative drawing, half of the edges point upwards

Figure 7. Different drawings of the same tree. The example illustrates the challenges faced when the width of a tree's drawing should be kept small. (a)–(c) are created with existing layering methods, (d) demonstrates what would be possible if a subset of the graph's edges were allowed to point upwards.

of the graph. In particular this means that the bound on the number of layers in the aforementioned integer program of Healy and Nikolov cannot be smaller than the longest path. We are not aware of any approach apart from the the integer program that aims at reducing the layering's height.

Trees. In this paper we refrain from specifically handling sub-trees of the graph and leave this important task to future work. In the following we shortly explain our motives. At first glance, the task to lay out a tree seems to be easy. Nevertheless, a significant amount of work has been devoted to forging layout algorithms for trees and to identifying criteria that make up a good drawing of a tree. A class of such algorithms is called *level-based*. The idea is to assign nodes with the same distance to the graph's root to the same level. Naturally, such drawings look similar to those of the layer-based approach. A drawback of the level-based algorithms is that the drawings become rather wide. An up-to-date summary of tree layout algorithms is given by Rusu [31].

A general directed graph can contain several sub-trees. Computing a layering for the whole graph with one of the layering methods discussed above thus computes layerings for the individual sub-trees as well. How well do the traditional layering methods perform for trees? The longest path algorithm would place all leaves of each sub-tree in the bottom-most layer, possibly elongating edges unnecessarily, and thus resulting in unfortunate drawings. The approach of Gansner et al. on the other hand assigns the layers in the same way level-based

tree algorithms would do, resulting in natural drawings of the sub-trees.

This being said, it is not immediately clear what a good way to re-arrange the nodes of a leveled drawing of a tree is to create a drawing with less width. Dedicated tree layout algorithms solve this problem, for instance, by placing the nodes on concentric circles (*radial layout*) or by alternating the layout direction for sub-trees (*horizontal-vertical layout*). Here, we are not able to move away from assigning nodes to layers since we do not want to alter the philosophy of the layer-based approach.

Nachmanson et al. explicitly mention the desire to reduce the fan-out of large trees in their paper but note that additional heuristics are required to produce acceptable layouts. Consider Figure 7. Drawings (b) and (c) are created with layering methods that specifically aim at smaller widths. While being slightly narrower, the overall drawing loses quality. In drawing (d) half of the graph's edges point upwards. As a result, a drawing of half of (a)'s width is possible. Still, the aforementioned problems are just the same for the "upward pointing" subgraph and the "downward pointing" subgraph.

From a theoretical point of view, the width of a tree's layering cannot be altered when dummy nodes contribute just as much to the width of a layer as original nodes. The width can be halved when edges are allowed to point into the "wrong" direction.

In practice it is hardly the case that dummy nodes contribute the same width as original nodes; dummy nodes represent edges, and edges are usually significantly narrower than nodes. Nevertheless, many coordinate assignment algorithms (phase 4) aim at placing consecutive dummy nodes at the same x coordinate, which is synonymous with the desire to have straight edges. As a consequence, they often space the dummy nodes further apart than theoretically necessary.

Another point to consider is that sub-trees at different places in the graph likely demand individual treatment based on the immediate neighborhood.

Force-directed. In the context of force-directed layout, Dwyer and Koren presented a method that can incorporate constraints enforcing all directed edges to point into the same direction [9]. It is also denoted as *constrained stress-minimizing layout*. They explored the possibility to relax some of the constraints, i. e. let some of the edges point backwards, and found that this improves the readability of the drawing. In particular, it reduced the number of edge crossings.

Building on constrained stress-minimization, Kieffer et al. present a layout method they call *human-like orthogonal layout (HOLA)* [22]. A central idea is to remove sub-trees from the input graph during a pre-processing step, lay them out separately using dedicated tree layout algorithms, and integrate them into the final drawing later on. We believe this is a promising strategy that should be explored when handling the subtrees during layering.

Disruptiveness. The idea behind seeking a minimum number of edges to reverse during cycle removal is to disturb the hierarchy represented by the graph as little as possible. The number of reversed edges therefore is a quality measure for the disruptive effect of the layout algorithm. However, it is not clear, and likely depends on the diagram type at hand, if the reversal of certain edges has less of a disruptive effect than the reversal of other edges. For instance, a graph may contain a central path that should not be interrupted. Likewise, it is not clear to what extend users accept a more compact drawing at the cost of a disturbed hierarchy. Both points are not easy to answer and require extensive research. Ideas from other areas may help here, e. g. Gupte et al. seek hierarchy in social networks [15] based on the term *social agony*.

A different question is whether or not two different drawings of the same graph, with different compactness for instance, look alike. Several terms are used within the graph drawing community to denote algorithms that build on existing drawings or exiting layout information to compute a new drawing, for instance: *layout adjustment* [24], *incremental layout* [27], and *interactive layout* [34]. All aim at *layout stability* and at preserving a user's *mental map* [10]. Consequently, metrics have been proposed to measure stability and mental map preservation, see Brake [3] for an initial overview. It remains to be evaluated which metrics are relevant and applicable when creating layerings.

In this paper we are first and foremost interested in a new layering method that is able to overcome current limitations. The higher the cost of reversed edges in GLP, the more the hierarchy of the graph is preserved in the drawing. However, GLP does not yet take subjective user perception or layout stability into account.

8 Conclusions

In this paper we address problems with current methods for the first two phases of the layer-based layout approach. We argue that separately performing cycle removal and layering is disadvantageous when aiming for compactness or specific drawing areas.

We present a configurable method for the layering phase that, compared to other state-of-the-art methods, shows on average improved performance on compactness. That is, the number of dummy nodes is reduced significantly for most graphs and can never increase. While the number of dummy nodes only allows for an estimation of the area, the effective area of the final drawing, measures in pixels, is reduced as well. Furthermore, graph instances for which current methods yield unfavorable aspect ratios can easily be improved. The performance of the presented heuristic is largely comparable to the optimal solutions delivered by the IP approach.

Building on these findings, we conduct experiments where we explicitly target a drawing area of certain aspect ratio, and find that our methods leave room for improvements when more edges may be reversed. We therefore suggest to address use cases where directionality is very important, and use cases

for which the direction is less important and an increased number of reversed edges is acceptable, with fundamentally different approaches.

Further, we want to stress that the common practice to determine the quality of methods developed for certain phases of the layer-based approach based on metrics that represent estimations of the properties of the final drawing is error-prone. For instance, estimations of the area and aspect ratio based on the results of the layering phase can vary significantly from the effective values of the final drawing and strongly depend on the used strategies for computing node and edge coordinates. Also, nodes can have significantly different sizes in both width and height; something we mentioned but not specifically addressed in this work.

We see several directions to proceed from here. For scenarios where directionality is less important, we plan to improve the presented heuristic, e. g. by selecting the initial node based on a certain criterion instead of randomly (see Algorithm 1), and by supporting (sub-)trees. For scenarios where directionality is important, we plan to further investigate our suggestion to bluntly cut traditional drawings (see Figure 6). It applies to both scenarios that certain diagram types demand certain edges to be drawn forwards. For this, methods must support to prevent, or at least to strongly penalize, the reversal of certain edges. Also, user studies could help understand which edges are natural candidates to be reversed from a human's perspective.

Acknowledgements

We thank Chris Mears and Adalat Jabrayilov for support formulating integer programs, and we thank Tim Dwyer and Petra Mutzel for valuable discussions.

References

[1] B. Berger and P. W. Shor. Approximation algorithms for the maximum acyclic subgraph problem. In *Proceedings of the first annual ACM-SIAM symposium on Discrete algorithms (SODA'90)*, pages 236–243. SIAM, 1990.

[2] U. Brandes and B. Köpf. Fast and simple horizontal coordinate assignment. In P. Mutzel, M. Jünger, and S. Leipert, editors, *Proceedings of the 9th International Symposium on Graph Drawing (GD'01)*, volume 2265 of *LNCS*, pages 33–36. Springer, 2002. doi:10.1007/3-540-45848-4.

[3] J. Branke. Dynamic graph drawing. In M. Kaufmann and D. Wagner, editors, *Drawing Graphs: Methods and Models*, volume 2025 of *LNCS*. Springer, 2001.

[4] C. Buchheim, M. Jünger, and S. Leipert. A fast layout algorithm for k-level graphs. In J. Marks, editor, *Proceedings of the 8th International Symposium on Graph Drawing (GD'00)*, volume 1984 of *LNCS*, pages 229–240. Springer, 2001. doi:10.1007/3-540-44541-2.

[5] P. Charbit, S. Thomassé, and A. Yeo. The minimum feedback arc set problem is np-hard for tournaments. *Combinatorics, Probability & Computing*, 16(1):1–4, 2007. doi:10.1017/S0963548306007887.

[6] E. G. Coffman. and R. L. Graham. Optimal scheduling for two-processor systems. *Acta Informatica*, 1(3):200–213, 1972. doi:10.1007/BF00288685.

[7] C. Demetrescu and I. Finocchi. Combinatorial algorithms for feedback problems in directed graphs. *Information Processing Letters*, 86(3):129–136, 2003. doi:10.1016/S0020-0190(02)00491-X.

[8] G. Di Battista, A. Garg, G. Liotta, A. Parise, R. Tamassia, E. Tassinari, F. Vargiu, and L. Vismara. Drawing directed acyclic graphs: An experimental study. In S. C. North, editor, *Proceedings of the Symposium on Graph Drawing (GD'96)*, volume 1190 of *LNCS*, pages 76–91. Springer, 1997. doi:10.1007/3-540-62495-3_39.

[9] T. Dwyer and Y. Koren. Dig-CoLa: directed graph layout through constrained energy minimization. In *Proceedings of the IEEE Symposium on Information Visualization (INFOVIS'05)*, pages 65–72, Oct. 2005. doi:10.1109/INFVIS.2005.1532130.

[10] P. Eades, W. Lai, K. Misue, and K. Sugiyama. Preserving the mental map of a diagram. In *Proceedings of the First International Conference on Computational Graphics and Visualization Techniques*, pages 34–43, 1991.

[11] P. Eades, X. Lin, and W. F. Smyth. A fast and effective heuristic for the feedback arc set problem. *Information Processing Letters*, 47(6):319–323, 1993. doi:10.1016/0020-0190(93)90079-O.

[12] P. Eades and K. Sugiyama. How to draw a directed graph. *Journal of Information Processing*, 13(4):424–437, 1990.

[13] G. Even, J. Naor, B. Schieber, and M. Sudan. Approximating minimum feedback sets and multicuts in directed graphs. *Algorithmica*, 20(2):151–174, 1998. doi:10.1007/PL00009191.

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

[15] M. Gupte, P. Shankar, J. Li, S. Muthukrishnan, and L. Iftode. Finding hierarchy in directed online social networks. In *Proceedings of the 20th International Conference on World Wide Web*, pages 557–566, 2011. doi:10.1145/1963405.1963484.

[16] C. Gutwenger, R. von Hanxleden, P. Mutzel, U. Rüegg, and M. Spönemann. Examining the compactness of automatic layout algorithms for practical diagrams. In *Proceedings of the Workshop on Graph Visualization in Practice (GraphViP '14)*, Melbourne, Australia, July 2014.

[17] P. Healy and N. S. Nikolov. A branch-and-cut approach to the directed acyclic graph layering problem. In S. G. Kobourov and M. T. Goodrich, editors, *Proceedings of the 10th International Symposium on Graph Drawing (GD'02)*, volume 2528 of *LNCS*, pages 98–109. Springer, 2002. doi:10.1007/3-540-36151-0.

[18] P. Healy and N. S. Nikolov. How to layer a directed acyclic graph. In P. Mutzel, M. Jünger, and S. Leipert, editors, *Proceedings of the 9th International Symposium on Graph Drawing (GD'01)*, volume 2265 of *LNCS*, pages 16–30. Springer, 2002. doi:10.1007/3-540-45848-4_2.

[19] P. Healy and N. S. Nikolov. Hierarchical drawing algorithms. In R. Tamassia, editor, *Handbook of Graph Drawing and Visualization*, pages 409–453. CRC Press, 2013.

[20] M. Hecht. Exact localisations of feedback sets. *Theory of Computing Systems*, May 2017. doi:10.1007/s00224-017-9777-6.

[21] A. Jabrayilov, S. Mallach, P. Mutzel, U. Rüegg, and R. von Hanxleden. Compact layered drawings of general directed graphs. In *Proceedings of the 24nd International Symposium on Graph Drawing and Network Visualization (GD '16), LNCS 9801*, pages 209–221, 2016. doi:10.1007/978-3-319-50106-2.

[22] S. Kieffer, T. Dwyer, K. Marriott, and M. Wybrow. HOLA: human-like orthogonal network layout. *IEEE Trans. Vis. Comput. Graph.*, 22(1):349–358, 2016. doi:10.1109/TVCG.2015.2467451.

[23] A. J. McAllister. A new heuristic algorithm for the linear arrangement problem. Technical report, University of New Brunswick, 1999.

[24] K. Misue, P. Eades, W. Lai, and K. Sugiyama. Layout adjustment and the mental map. *Journal of Visual Languages & Computing*, 6(2):183–210, June 1995. doi:10.1006/jvlc.1995.1010.

[25] L. Nachmanson, G. Robertson, and B. Lee. Drawing graphs with GLEE. In S.-H. Hong, T. Nishizeki, and W. Quan, editors, *Graph Drawing*, volume 4875 of *LNCS*, pages 389–394. Springer Berlin Heidelberg, 2008. doi:10.1007/978-3-540-77537-9_38.

[26] N. S. Nikolov, A. Tarassov, and J. Branke. In search for efficient heuristics for minimum-width graph layering with consideration of dummy nodes. *Journal of Experimental Algorithmics*, 10, 2005. doi:10.1145/1064546.1180618.

[27] S. C. North. Incremental layout in DynaDAG. In *Proceedings of the Symposium on Graph Drawing*, volume 1027 of *LNCS*, pages 409–418. Springer, 1996. doi:10.1007/BFb0021824.

[28] J. Pantrigo, R. Martí, A. Duarte, and E. Pardo. Scatter search for the cutwidth minimization problem. *Annals of Operations Research*, 199(1):285–304, 2012. doi:10.1007/s10479-011-0907-2.

[29] U. Rüegg, T. Ehlers, M. Spönemann, and R. von Hanxleden. A generalization of the directed graph layering problem. Technical Report 1501, Kiel University, Department of Computer Science, Feb. 2015. ISSN 2192-6247.

[30] U. Rüegg, T. Ehlers, M. Spönemann, and R. von Hanxleden. A generalization of the directed graph layering problem. In *Proceedings of the 24th International Symposium on Graph Drawing and Network Visualization (GD '16), LNCS 9801*, pages 196–208, 2016. doi:10.1007/978-3-319-50106-2_16.

[31] A. Rusu. Tree drawing algorithms. In R. Tamassia, editor, *Handbook of Graph Drawing and Visualization*, pages 155–192. CRC Press, 2013.

[32] G. Sander. A fast heuristic for hierarchical Manhattan layout. In F. J. Brandenburg, editor, *Proceedings of the Symposium on Graph Drawing (GD'95)*, volume 1027 of *LNCS*, pages 447–458. Springer, 1996. doi:10.1007/BFb0021828.

[33] G. Sander. Layout of directed hypergraphs with orthogonal hyperedges. In G. Liotta, editor, *Proceedings of the 11th International Symposium on Graph Drawing (GD'03)*, volume 2912 of *LNCS*, pages 381–386. Springer, 2004. doi:10.1007/978-3-540-24595-7_35.

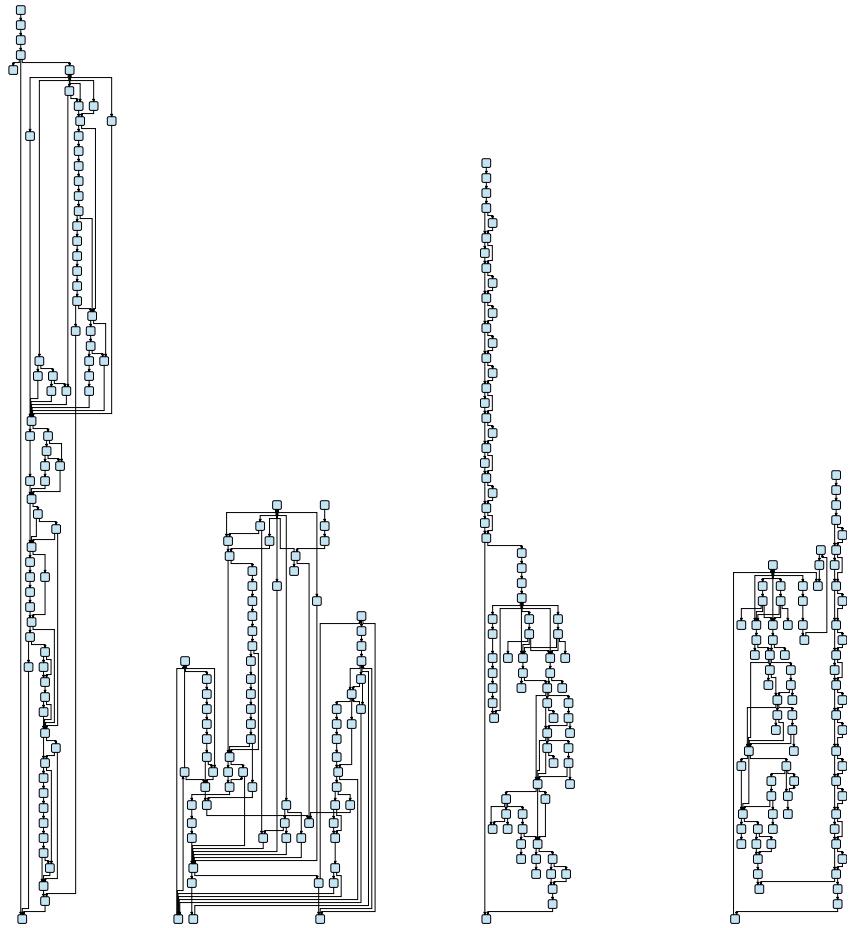
[34] M. Spönemann. *Graph layout support for model-driven engineering*. Number 2015/2 in Kiel Computer Science Series. Department of Computer Science, 2015. Dissertation, Faculty of Engineering, Christian-Albrechts-Universität zu Kiel.

[35] K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierarchical system structures. *IEEE Transactions on Systems, Man and Cybernetics*, 11(2):109–125, Feb. 1981.

[36] A. van Zuylen and D. P. Williamson. Deterministic pivoting algorithms for constrained ranking and clustering problems. *Mathematics of Operations Research*, 34(3):594–620, 2009. doi:10.1287/moor.1090.0385.

A Appendix

A.1 Example Drawings of North Graphs



(a) g.75.1 **(b)** g.80.5

Figure 8. For each graph the left drawing is produced using EaGa and the right drawing using the GLP-H as presented here. Graphs are taken from the North graphs library.

(a) g.99.0 **(b)** g.86.5

Figure 9. For each graph the left drawing is produced using EaGa and the right drawing using the GLP-H heuristic as presented here. Graphs are taken from the North graphs library.

A.2 Detailed Results

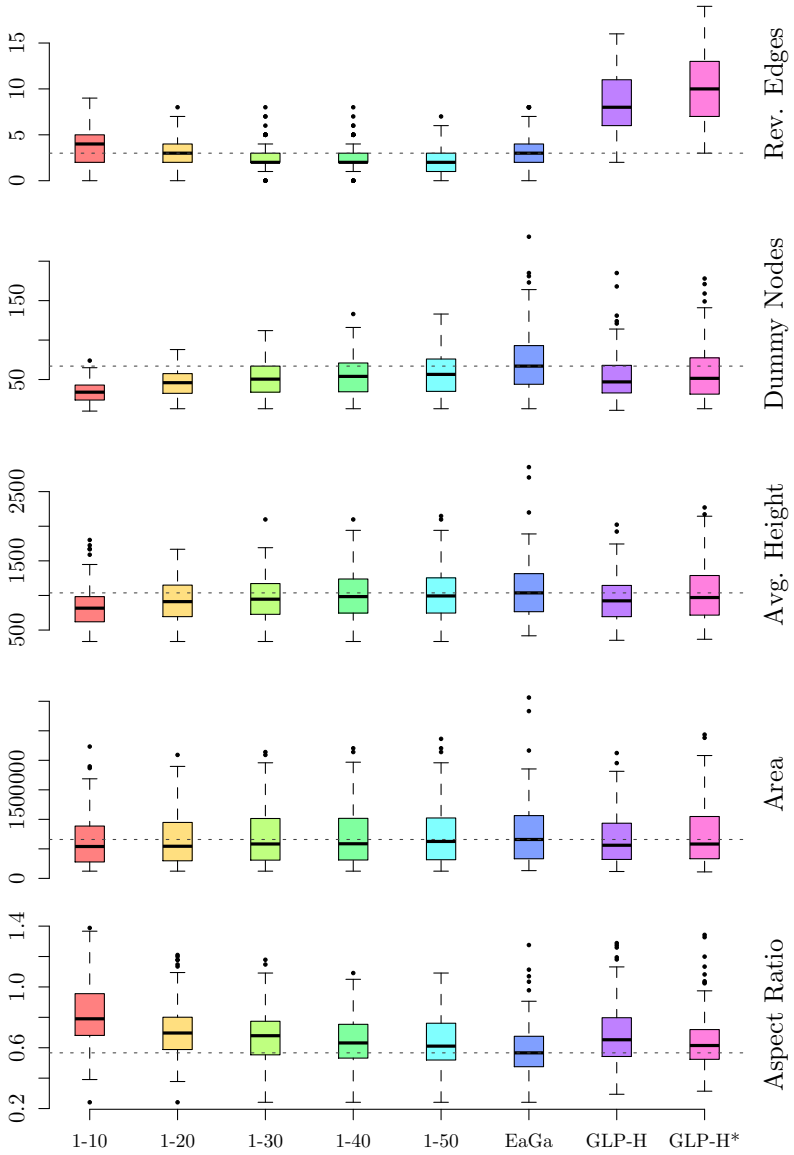


Figure 10. Random graphs: Detailed results in the form of boxplots. A summary can be seen in Table 1a. The dashed line represents the median of EaGa. Lower values are better, with the exception of the aspect ratio. It can be seen that the methods presented here, improve the drawing w. r. t. the relevant metrics. It is noteworthy that for 1-30-GLP, 1-40-GLP, and 1-50-GLP, both the number of reversed edges and the number of dummy nodes is smaller compared to EaGa.

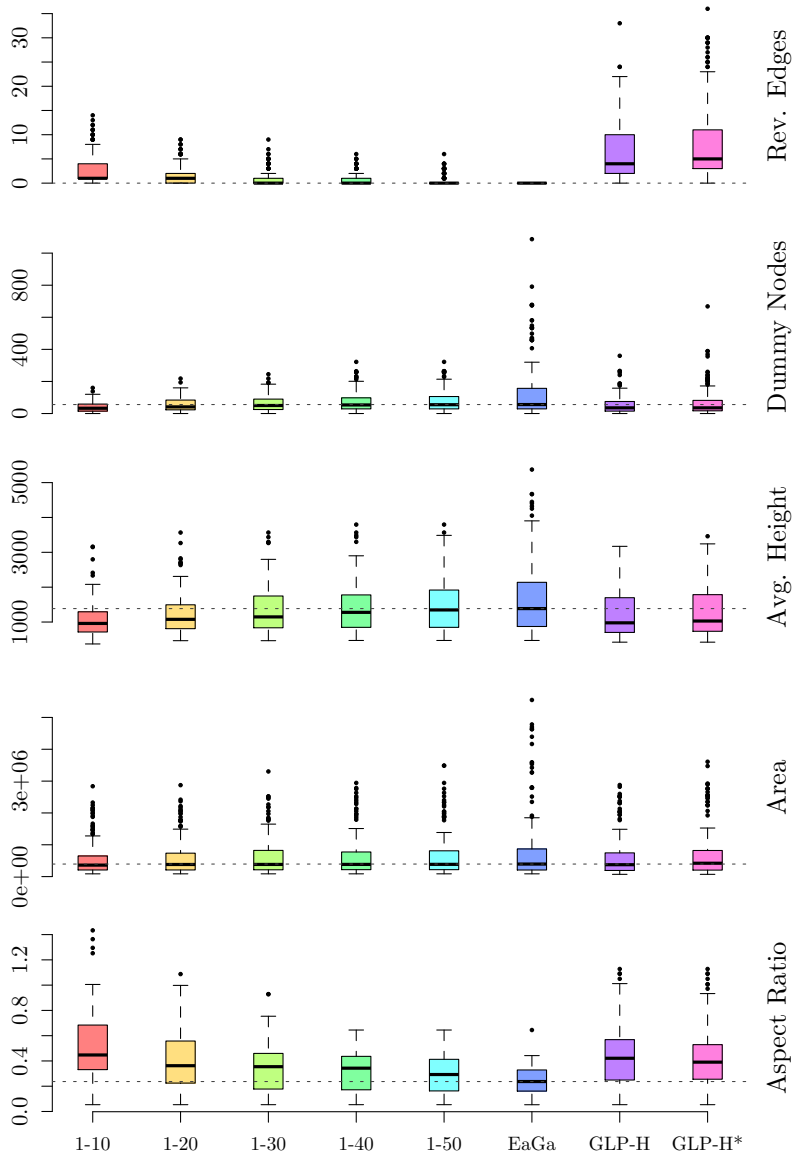


Figure 11. North graphs: Detailed results in the form of boxplots. A summary can be seen in Table 1b. The dashed line represents the median of EaGa. Lower values are better, with the exception of the aspect ratio. It can be seen that the methods presented here, improve the drawing w. r. t. the relevant metrics. The North graphs are acyclic which is why EaGa consistently produces zero reversed edges. The aspect ratio of the majority of the graphs improves significantly with a constant to slightly improved area.

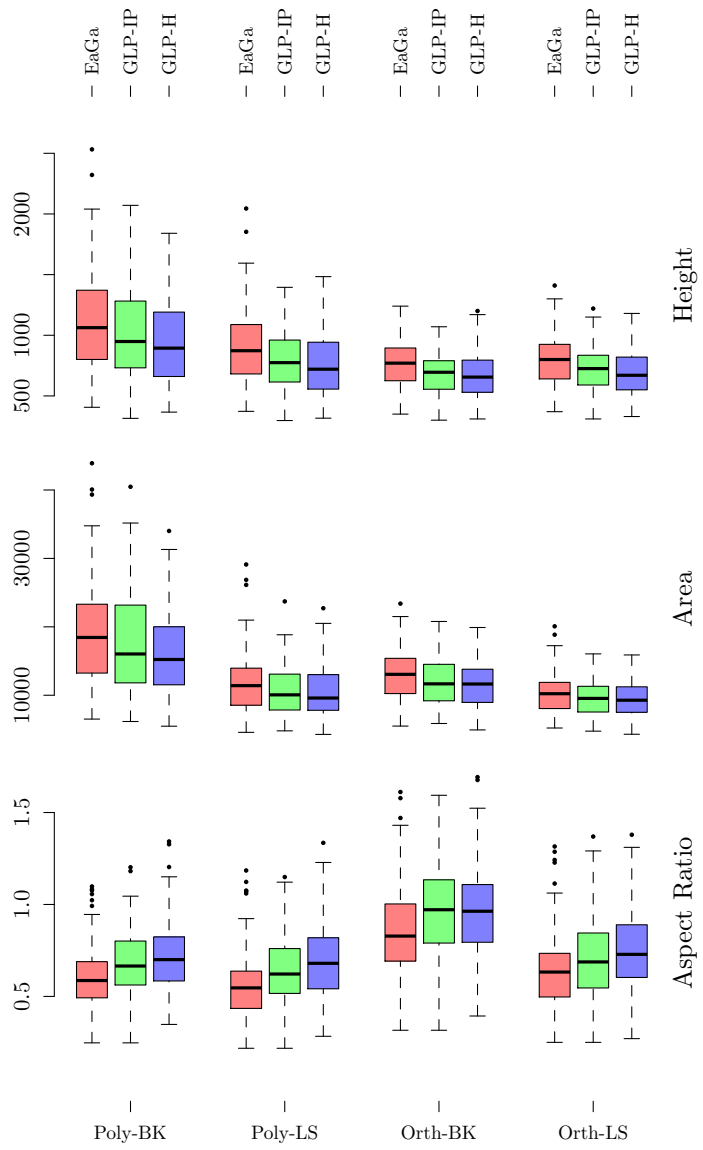


Figure 12. Detailed results for the produced drawings using different strategies for the layer-based approach's phases as discussed in Section 5 (cf. Table 2). It can be seen that for every combination GLP-IP and GLP-H improve w. r. t. the tested metrics when compared to EaGa. Furthermore, the results emphasize that different strategies can result in significantly different drawings, especially when it comes to aspect ratio. For instance, orthogonal-style edges allow for less height and area. Node coordinates assigned by LS tend to allow for smaller area than BK.

A.3 Full GLP-MS MIP Model

The model is a modification of the CGL formulation as introduced by Jabrayilov et al. [21]. Given is a graph $G = (V, E)$, which includes a dummy edge $(s, e) \in E$, as discussed in Section 6.1. Let $H = |V|$ and let r_w and r_h denote the input parameters specifying the drawing area. The following binary variables are used: $y_{v,k}$ equals 1 if and only if $L(v) < k$. Note that the reverse variables $y_{k,v}$ are part of the model for ease of understanding but can be eliminated when implementing the model. $r_{u,v}$ equals 1 if and only if the edge (u, v) is reversed. $z_{uv,k}$ equals 1 if and only if the edge (u, v) produces a dummy node in layer k.

$$\text{Minimize } \big(\omega_{rev} \sum_{(u,v)\in E} r_{u,v}\big) + \big(\omega_{len} \sum_{(u,v)\in E} \sum_{k=2}^{H-1} z_{uv,k}\big) + \omega_{\text{ms}}\, MS, \;\; s.t.$$

$y_{v,1}$	$= 0$	for all $v \in V$	(H)
$y_{H,v}$	$= 0$	for all $v \in V$	(I)
$y_{k,v} + y_{v,k+1}$	$= 1$	for all $v \in V, 1 \le k \le H-1$	(J)
$y_{k+1,v} - y_{k,v}$	≤ 0	for all $v \in V, 1 \le k \le H-2$	(K)
$-y_{u,k} - y_{k,v} - r_{u,v}$	≤ -1	for all $(u,v) \in E, 1 \le k \le H$	(L)
$-y_{k,u} - y_{v,k} + r_{u,v}$	≤ 0	for all $(u,v) \in E, 1 \le k \le H$	(M)
$y_{k,u} + y_{v,k} - z_{uv,k}$	≤ 1	for all $(u,v) \in E, 2 \le k \le H-1$	(N)
$y_{k,v} + y_{u,k} - z_{uv,k}$	≤ 1	for all $(u,v) \in E, 2 \le k \le H-1$	(O)
$y_{k,s}$	$= 0$	for all $1 \le k \le H$	(P)
$y_{k,v}$	$\le y_{k,e}$	for all $1 \le k < H, v \in V \setminus \{e\}$	(Q)
$\dfrac{\left(\sum_{u\in V}(1 - y_{u,k} - y_{k,u})\right) - 1}{r_w}$	$\le MS$	for all $k \in \{1, H\}$	(R)
$\dfrac{\left(\sum_{u\in V}(1 - y_{u,k} - y_{k,u}) + \sum_{(u,v)\in E} z_{uv,k}\right) - 1}{r_w}$	$\le MS$	for all $2 \le k \le H-1$	(S)
$\dfrac{1 + \sum_{1\le k\le H} y_{k,e}}{r_h}$	$\le MS$		(T)
$y_{v,k},\ y_{k,v}$	$\in \{0,1\}$	for all $v \in V, 1 \le k \le H$	
$r_{u,v}$	$\in [0,1]$	for all $(u,v) \in E$	
$z_{uv,k}$	$\in [0,1]$	for all $(u,v) \in E, 2 \le k \le H-1$	
MS	$\in \mathbb{R}_{\ge 0}$		