INSTITUT FÜR INFORMATIK

Minimum-Width Graph Layering Revisited

Ulf Rüegg, Marc Adolf, Michael Cyruk,
Astrid Mariana Flohr, and Reinhard von Hanxleden

Bericht Nr. 1701
February 2017
ISSN 2192-6247

CHRISTIAN-ALBRECHTS-UNIVERSITÄT

ZU KIEL

Department of Computer Science
Kiel University
Olshausenstr. 40
24098 Kiel, Germany

Minimum-Width Graph Layering Revisited

Ulf Rüegg, Marc Adolf, Michael Cyruk,
Astrid Mariana Flohr, and Reinhard von Hanxleden

Report No. 1701
February 2017
ISSN 2192-6247

E-mail: {uru,mad,mic,amf,rvh}@informatik.uni-kiel.de

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

Abstract

The minimum-width layering problem tackles one step of a layout pipeline to create top-down drawings of directed acyclic graphs. Thereby, it aims at keeping the overall width of the drawing small.

We study layering heuristics for this problem as presented by Nikolov et al. as well as traditional layering methods with respect to the questions how close they are to the true minimal width and how well they perform in practice, especially, if a particular drawing area is prescribed.

We find that when applied carefully and at the right moment the layering heuristics can, compared to traditional layering methods, produce better layerings for prescribed drawing areas and for graphs with varying node dimensions. Still, we also find that there is room for improvement.

Keywords: layer-based layout, layer assignment, minimum-width layering, fixed drawing area

Contents

1 Introduction

The *layer-based layout approach* is a widely used method to automatically draw directed graphs and was introduced by Sugiyama et al. in 1981 [13]. It is based on the idea to assign nodes to subsequent *layers* with as many edges pointing into the same direction as possible in order to emphasize the inherent direction of the graph. See Figure 1.1 for an example. The literature's common layout direction is top-down, i. e. layers are horizontal strips and edges point downwards. The method splits the layout task into three consecutive phases: (1) the *node layering* phase distributes the nodes into indexed layers such that all edges point from layers with lower index to layers with higher index, edges that span multiple layers are split using *dummy nodes* such that every edge connects nodes in adjacent layers, (2) the *crossing minimization* phase orders the nodes in each layer such that the number of edge crossings is minimized, and finally (3) the *node placement* phase determines the actual node coordinates. In practice, an initial *cycle breaking* phase as well as a final *edge routing* phase are often added to support cyclic graphs and different edge routing styles.

In this report, we are interested in the layering phase, in particular, in a specialized version of it, which is referred to as *minimum-width layering*. One seeks to produce layerings which have a small *width*, where the width of a layering is defined as the maximum number of nodes in any layer of the layering. One can either consider the contribution of dummy nodes to the width of a layer or not. Unless stated otherwise we do consider dummy nodes. The *height* of the layering is defined as the number of used layers.

Branke et al. showed that the problem of finding a layering with minimum width is NP-complete when dummy nodes are considered [1]. Healy and Nikolov presented an ILP approach to minimize the number of dummy nodes in a layering subject to bounds on the width and height of the layering [7]. In a subsequent paper they solve their presented ILP using a branch and cut algorithm which significantly decreases the execution time [6]. In further papers Nikolov, Tarassov, and Branke present several heuristics to solve the minimum-width layering problem [14, 11] and to reduce the number of dummy nodes in the resulting layerings [10]. These heuristics are the main subject of research in this report.

Contrary to the layering methods just mentioned, traditional layering methods have no control over the width of the resulting layering. The well-known *longest path algorithm (LP)* guarantees a layering with the minimum number of layers [3], and the *network simplex approach (NS)* presented by Gansner et al. guarantees a minimum number of dummy nodes [5]. Both approaches, however, can result in layerings with an unfortunate width. Adapting Coffman and Graham's scheduling algorithm with precedence constraints to the layering problem, one can restrict the number of regular nodes per

layer [2]. However, it does not allow to consider the contribution of dummy nodes, which can have a significant impact on the width of a drawing.

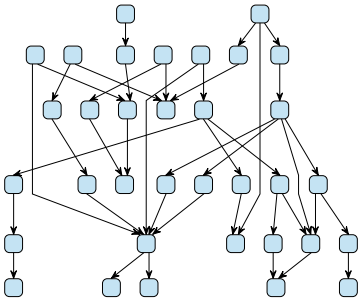
Contributions. In this report, we answer the following questions:

1. While the problem we are concerned with is called minimum-width layering, the heuristics of Nikolov et al. do not necessarily find a layering with minimal width. After all they are heuristics. What is the gap between heuristic solutions and optimum solutions? How do the traditional layering methods compare to optimal solutions? Both points were not yet examined by Nikolov et al.

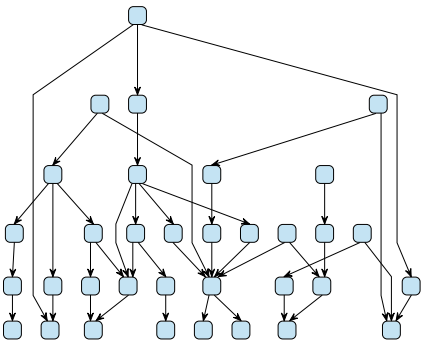
2. Can the heuristics be used to target a certain drawing area, for instance a computer screen?

3. Is there a need for and an easy way to extend the heuristics to consider differently-sized nodes?

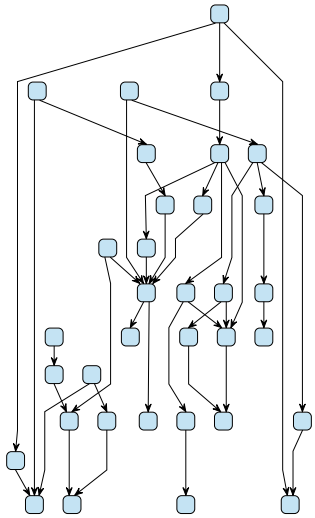
Outline. We proceed by introducing necessary terminology and the set of graphs used for our evaluations in the next chapter. Afterwards we tackle Questions 1–3 in consecutive sections of Chapter 3. The final chapter contains summarizing discussions.



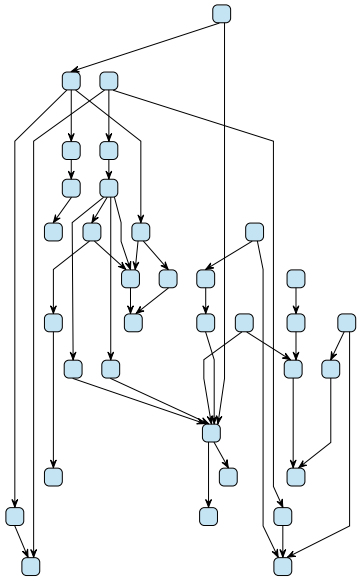
(a) NS



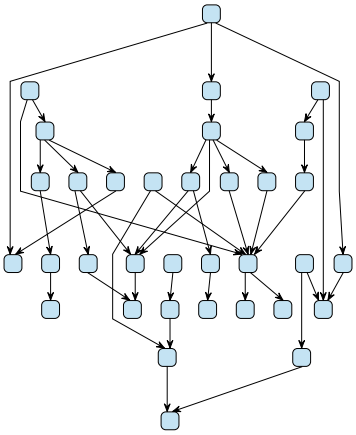
(b) LP



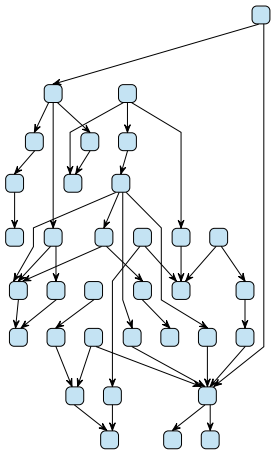
(c) MinWidth



(d) StretchWidth



(e) MinWidth and PromoteNodes



(f) StretchWidth and PromoteNodes

Figure 1.1: Different layerings of the grafo11465.34 graph from the Rome graphs collection, produced with traditional methods, (a) and (b), and with the heuristics of Nikolov et al. [11], (c)–(f).

2 Preliminaries

This chapter introduces the terminology used throughout this report.

Definition 2.1. Let $G = (V, E)$ be a graph with a set of nodes V and a set of directed edges E. A *layering* of G is a partition of V into disjoint subsets $\mathcal{L} = \{L_1, \ldots, L_{|\mathcal{L}|}\}$.

Definition 2.2. Let $G = (V, E)$ be a directed graph. For a node $v \in V$, the *in-degree* $d^-(v)$ is the number of directed edges ending in v and the *out-degree* $d^+(v)$ is the number of edges leaving v. $\bar{d}^-(G)$ denotes the average in-degree of all nodes in G and $\bar{d}^+(G)$ the average out-degree.

Definition 2.3. Let $G = (V, E)$ be a directed acyclic graph. A layering $\mathcal{L}$ is *valid* if $\forall (u, v) \in E$ with $u \in L_i \wedge v \in L_j$: $i < j$.

An edge $(u, v) \in E$ is *short* in a valid layering if $j - i = 1$ for $u \in L_i \wedge v \in L_j$, otherwise it is *long*. A valid layering L is *proper* if all edges are short. Given any valid layering, it can be made proper by splitting long edges with sequences of *dummy nodes*. This gives a graph $G' = (V \cup D, E')$, where D is the set of dummy nodes and E' contains only short edges. A node that is not a dummy node is referred to as *regular node*.

Definition 2.4. A node of a graph G may have a width $w(v)$ and a height $h(v)$. The width of the edge representing dummy nodes is denoted by w^d.

2.1 Measures

The introduction includes definitions of the width and the height of a layering that have been used in the literature before. However, they represent estimates that can deviate from the actual dimensions of a final drawing, e. g. in pixels when drawn on a screen. Therefore, we define these terms more precisely in the following.

Definition 2.5. For a graph G with layering $\mathcal{L}$, the *estimated height* $\widetilde{h}$ of the layering is the number of layers $|\mathcal{L}|$ and the *estimated width* $\widetilde{w}$ is the maximum number of nodes in a layer over all layers: $\max_{L \in \mathcal{L}} |L|$. A properly layered graph may include dummy nodes. Unless stated otherwise we do include the dummy nodes in $\widetilde{w}$.

Definition 2.6. Let G be a graph with a layering $\mathcal{L}$. The *estimated area* $\widetilde{a}$ is the estimated width of $\mathcal{L}$ multiplied by its estimated height. The *estimated aspect ratio* $\widetilde{ar}$ is the quotient of estimated width and estimated height.

Definition 2.7. Let G be a graph with a layering $\mathcal{L}$. The *edge density* $\mathcal{E}(L_i)$ of a layer L_i is the number of edges spanning from layer L_i to layer L_{i+1}. The edge density of the last layer is thus always 0. The maximum edge density $\hat{\mathcal{E}}$ of the layering $\mathcal{L}$ is $\max_{1 \leq i < |\mathcal{L}|} \mathcal{E}(L_i)$. The average edge density $\bar{\mathcal{E}}$ is $\frac{1}{|\mathcal{L}|-1} \sum_{1 \leq i < |\mathcal{L}|} \mathcal{E}(L_i)$. If G is properly layered the average edge density can alternatively be stated as $\frac{1}{|\mathcal{L}|-1} (|E| + |D|)$, where E denotes the set of original edges and D denotes the set of introduced dummy nodes.

Definition 2.8. Let G be a graph. A *drawing* D of G is an embedding in the plane.

Definition 2.9. Let D be a drawing of a graph G. The *(effective) width* w of D is the difference between the maximal and the minimal horizontal coordinate component of any point in the embedding D. The *(effective) height* h is defined symmetrically for the vertical components.

Definition 2.10. Let D be a drawing of a graph G. The *effective area* a is the product of the width and the height of D. The *effective aspect ratio* ar is the quotient of width and height.

Definition 2.11. Let $\mathcal{R}_{(r_w, r_h)}$ denote a reference frame with width r_w and height r_h. For instance, an A4 paper sheet (portrait) has $\mathcal{R}_{(210\text{mm},\, 297\text{mm})}$, or simplified $\mathcal{R}_{(1,\sqrt{2})}$. The *max scale value* s of a drawing D with dimensions w and h with respect to a certain reference frame $\mathcal{R}$ is defined as:

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

Intuitively it represents the scaling factor by which all elements of a drawing (alongside their positions) have to be scaled in order to fit into the reference frame.

Definition 2.12. Given two drawings D and D', and their corresponding max scale values s and s' in relation to a common reference frame, 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 reference frame. In other words, if $r > 1$, the drawing D can be displayed larger than D'.

2.2 Test Graphs

Nikolov et al. used a subset of the *Rome graphs*[1] for their evaluations. From the original 11.530 graphs they removed graphs that contain directed cycles or are unconnected, leaving 5911 graphs. We applied the same criteria and ended up with 5912 graphs. However, the graph grafo7417.39 contains 104 nodes. For all other graphs the last number of the filename represents the number of nodes in the graph. Further, Nikolov et al. state their graphs contain between 10 and 100 nodes. Therefore, we removed this graph as well, leaving 5911 graphs. We refer to this set as RomeF during the remainder

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

of this report. The graphs of the set have between 10 and 100 nodes, 48.4 on average, and between 9 and 158 edges, 62.7 on average.

Before we continue to explain the next set of graphs, a word on the layout direction. As mentioned during the introduction, the prevalent layout direction in the literature of the layer-based layout approach is top-down. However, not all diagram types must be drawn top-down; a left-to-right layout is just as justified. We will explain our reasons in more detail in Section 3.3.1.

From the RomeF set we extracted three further subsets as follows. We laid out the RomeF graphs from left-to-right using layering and node placement techniques presented by Gansner et al. [5] and a simple polyline edge routing. For the resulting drawings we measured the effective aspect ratio and composed three sets of 1000 graphs each:

Low: 1000 graphs with the lowest aspect ratios (roughly 0.3–1.0)

Middle: 1000 graphs with aspect ratios around 1.6, the aspect ratio of up-to-date computer screens (roughly 1.5–1.7)

High: 1000 graphs with the highest aspect ratios (roughly 2.2–7.1)

The rationale for this partitioning is that we expect the layering heuristics of Nikolov et al. to perform differently based on the aspect ratio of a "traditional" drawing. Consider the following example: a low aspect ratio value of 0.5 in a left-to-right drawing may be an indication that the maximum of the number of nodes in any layer is larger than the number of layers. To increase the aspect ratio, thus better matching a computer screen, the maximum number of nodes per layer would have to be decreased by moving some nodes to new layers. Exactly this is what the heuristics do. More on this topic in Section 3.3.1.

The last set of graphs we introduce here has, as opposed to the Rome graphs, nodes with varying dimensions. In Section 3.4 we use it to evaluate our extensions of the heuristics to consider individual node sizes. *Sequentially Constructive Graphs (SCGs)* are specialized control flow graphs used in the context of *Sequentially Constructive Charts (SCCharts)* [15]. An example SCG can be seen in Figure 3.8 on page 16. Note that SCGs are drawn top-down with orthogonal edges. We assembled a set of 40 SCGs with aspect ratios between 1.3 and 4.5 and an outlier of 8.5. The graphs have between 32 and 563 nodes, 84.3 on average. On average of 9.2 nodes per graph are *hierarchical* nodes, i. e. nodes that contain further nodes. The edge count is between 40 and 827, 113.8 on average.

3 Minimum-Width Layering

Nikolov et al. were the first to introduce heuristics that create layerings with restricted width while considering dummy nodes [11]. In their 2005 paper they discuss the following three heuristics: MinWidth (MW), StretchWidth (SW), and PromoteNodes (PN). In this chapter we first give a brief introduction of the heuristics and their main differences. For an in-depth discussion we refer the reader to one of the original papers [10, 11, 14]. Second, we reconstruct evaluations of Nikolov et al., and present our contributions third. For the moment, the layout direction is top-down again.

MinWidth. The actions of the MW heuristic are roughly based on the longest path algorithm. The longest path algorithm iteratively places all sinks of a graph in a layer, removes the sinks from the graph, and continues with a new layer and the newly created sinks until the graph is empty. MW adds two new elements to this procedure: 1) instead of adding all available sinks to the current layer, it only adds nodes as long as a certain threshold on the width (and the predicted width of future layers) is not exceeded, and 2) the next node to be added to the current layer is chosen to be the one with the largest out-degree. The idea behind the second point is that the node with the largest out-degree has the most connections to already placed nodes; selecting it keeps the number of dummy nodes to be introduced low.

MW makes use of two input parameters. The authors conducted extensive parameter studies and found a set of eight promising parameter combinations. They propose to run the algorithm for all eight combinations and select the layering with smallest width.

StretchWidth. The second layering heuristic, SW, constructs the layering in a similar fashion to MW but does not depend on input parameters. Instead, SW starts with a low value for the threshold on the width of a layer and increases it iteratively whenever it finds that the threshold cannot be realized.

PromoteNodes. Nikolov et al. found that the two previously described heuristics tend to produce layerings with a large number of unnecessary dummy nodes, the removal of which would not necessarily harm the width of the layerings. They therefore suggest to execute a post-processing algorithm: PN. PN takes a given layering as input. It then recursively checks whether moving groups of nodes to earlier layers decreases the overall number of dummy nodes. When used in conjunction with MW or SW, nodes are only promoted if the width of the layering is not increased.

Differences. We take a closer look at the differences between MW and SW. The most obvious difference is that MW uses two input parameters (in addition to the graph itself) while SW does not use any input parameters. Apart from that both heuristics are based on the idea of the longest path layering algorithm and essentially have two further decisions to make: (1) select the node which should be placed next, and (2) start a new layer when a certain width is reached.

Regarding (1), out of all nodes that could possibly be placed in the current layer, MW picks the node v with the highest out-degree $d^+(v)$. SW on the other hand picks the node v with the highest *rank*, which is defined as $\max\{d^+(v), \max_{(u,v)\in E} \ d^+(u)\}$. The rank allows to look at nodes that cannot be placed yet but will have a significant influence on the layering's width as soon as they are placed. At the same time it yields shorter edges. Essentially, the rank realizes a one node lookahead with relation to the nodes' degrees.

Regarding (2), both heuristics internally use two variables: the width w_c of the currently constructed layer and an estimation of the width of future layers w_u. MW compares its two input parameters with these internal variables and starts a new layer if (a) w_c exceeds the input parameter for the upper bound on a layer's width *ubw* and no node with outgoing edges can be placed, or if (b) w_u exceeds *ubw* multiplied by the second input parameter *ubc*. Note that therefore *ubw* is not a bound on the width in a strict sense, as it allows further nodes to be placed in the current layer as long as they have outgoing edges. SW on the other hand starts with a lower bound on the allowed width w_c, for instance the maximum out-degree of any node in the graph, and increases this bound until it can construct a feasible layering. A new layer is created if either (a) w_c exceeds the current lower bound or if (b) w_u exceeds the current lower bound multiplied by the average out-degree of all of the graph's nodes.

MW first assigns a selected node to the current layer and then checks the condition if a new layer should be started. SW proceeds vice versa, first checking its condition and potentially starting a new layer, then assigning the selected node.

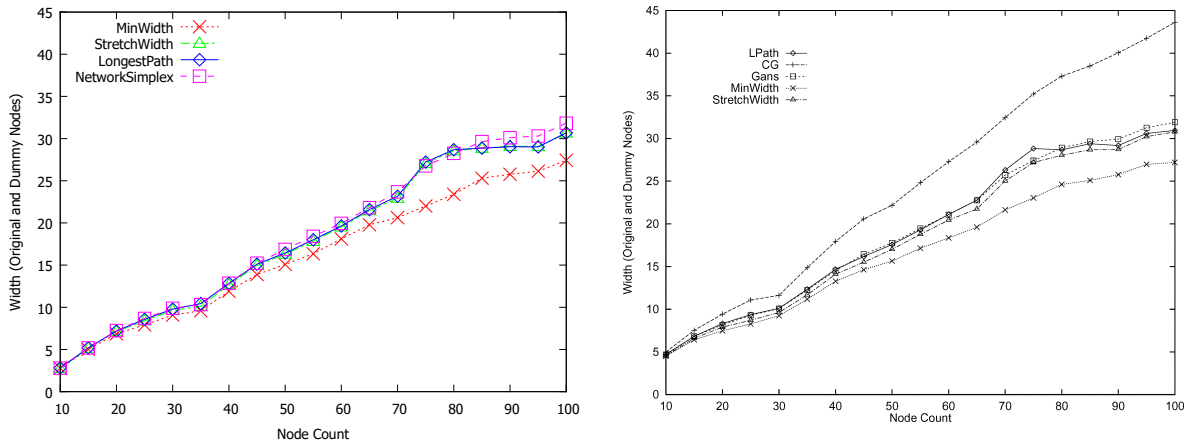
3.1 Reproducing Existing Results

We implemented the three heuristics as part of the ELK Layered algorithm of the Eclipse Layout Kernel (ELK), a Java-based open source project.[1]

As a first step we wanted to know if our implementations produce the same results as included in the paper of Nikolov et al. [11]. For this, we applied the heuristics to the RomeF graphs (cf. Section 2.2), plotted the results, and compared our plots to the plots of the original paper. An example illustrating the comparison of the plots for the estimated width with consideration of dummy nodes after node promotion can be seen in Figure 3.1. Note that our plot does not include the Coffman-Graham algorithm, which was the weakest one in the original evaluations [11].

We found that our implementations resemble the results of Nikolov et al. with very few and small deviations that are most likely either due to implementation details or

[1] `http://www.eclipse.org/elk`



(a) **(b)**

Figure 3.1: Plots for the estimated width with consideration of dummy nodes after node promotion. (a) shows our results, (b) the results as published by Nikolov et al. [11]. We did not implement the Coffman-Graham algorithm, which therefore is missing in (a).

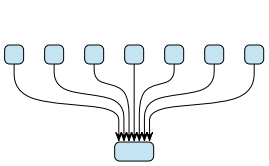
due to partly unspecified iteration orders of the nodes. As a consequence, we omit the plots comparing the remaining metrics: estimated height, number of dummy nodes, and edge density.

A Remark on Edge Density. According to Healy and Nikolov, drawings with a small maximum and average edge density are favorable [9]. They, however, give no proof for their claim. Consider Figure 3.2. Both diagrams have the same maximum edge density $\hat{\mathcal{E}}$ and the right diagram's average edge density $\bar{\mathcal{E}}$ is lower. One can argue though that the left diagram is more readable, at least when it comes to quickly understanding the graph's hierarchy.

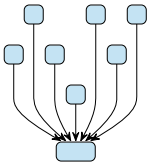
The maximum edge density represents the situation between a single pair of layers and completely neglects the structure of the rest of the layering. Also, it is bounded from below by either the maximum in-degree or the maximum out-degree of a graph's nodes. A single high-degree node can thus impair an otherwise totally fine layering.

The average edge density tends to be smaller for layerings with a small number of dummy nodes. This gives rise to the question why not simply use the number of dummy nodes as a measure. Moreover, the average edge density only depends on the number of layers for layerings with the minimum number of dummy nodes. As seen in Figure 3.3, the number of used layers in this case is not unique though. One can also both increase and decrease the average edge density of a given layering by inserting empty layers. In particular, it is possible to continuously decrease the average edge density by inserting more and more layers at the right place. This is demonstrated in Figure 3.4.

Since we are rather skeptical concerning the usefulness of the edge density as a quality measure we do not include it throughout our evaluations.

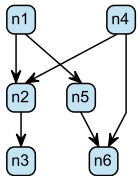


(a) $\hat{\bar{\varepsilon}} = 7$; $\bar{\bar{\varepsilon}} = 7$

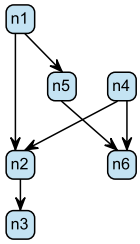


(b) $\hat{\bar{\varepsilon}} = 7$; $\bar{\bar{\varepsilon}} = 5.\bar{3}$

Figure 3.2: Both diagrams have the same maximum edge density but (b) has a lower average edge density. While a lower edge density is believed to characterize more comprehensible diagrams [9], (a) is arguably more comprehensible, when it comes to understanding the hierarchy.

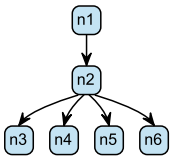


(a) $|D| = 1$; $|\mathcal{L}| = 3$

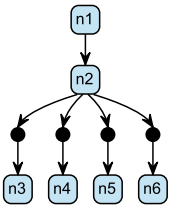


(b) $|D| = 1$; $|\mathcal{L}| = 4$

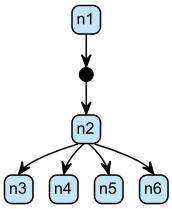
Figure 3.3: Both diagrams have one dummy node, which is the minimum number of dummy nodes possible, but (a) has only three layers while (b) has four. The graph (b) has been presented before by Healy and Nikolov to illustrate that dummy nodes can be part of the layering's longest path [8].



(a) $\bar{\bar{\varepsilon}} = 2.5$



(b) $\bar{\bar{\varepsilon}} = 3$



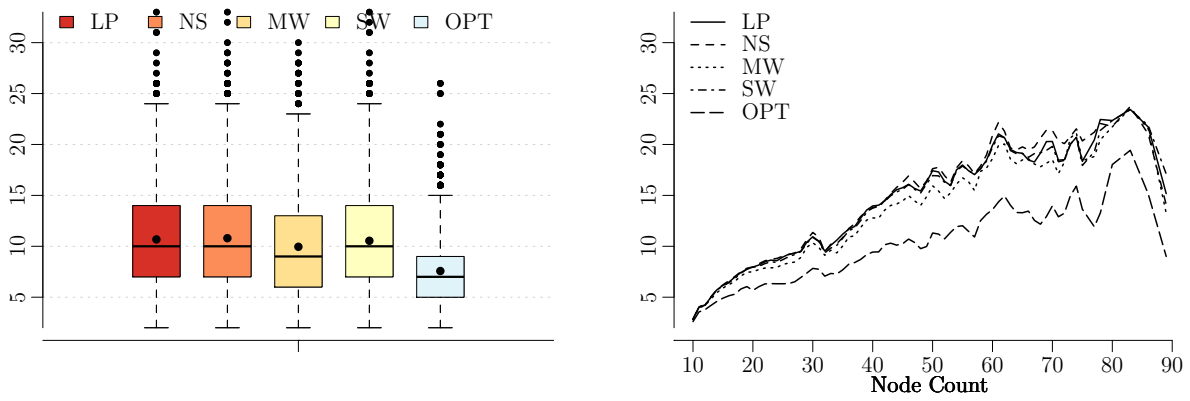
(c) $\bar{\bar{\varepsilon}} = 2$

Figure 3.4: The average edge density of a given drawing (a) can be altered by artificially inserting layers. The black circles represent dummy nodes. The average edge density of (b) is higher; as one would expect for that particular drawing. The average edge density of (c) is lower and converges to 1 if one continues to insert layers between `n1` and `n2`.

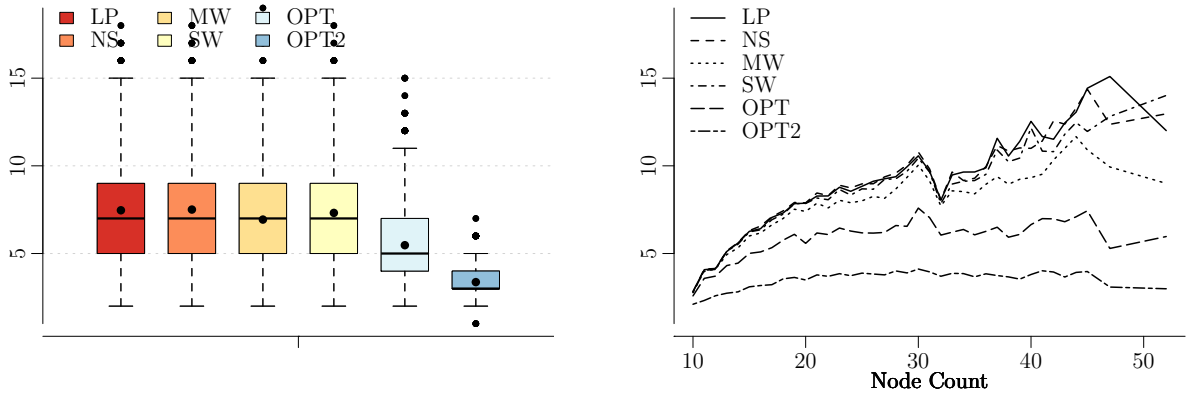
3.2 Truly Minimum Width

The heuristics of Nikolov et al. aim at finding layerings with minimal width, however, the authors never compare the widths of their layerings to the optimal solutions. We used an optimization problem, described in the next section, to compute a layering with minimal width (OPT). Out of interest, we dropped the requirement that edges have to point into a common direction and also computed the minimal width of this variation (OPT2). Results can be seen in Figure 3.5. As to the boxplots, the dot within the box denotes the average value, while the dots outside the whiskers denote outliers. They were created with *R 3.2.3*. In (a) the results of 3519 graphs are shown for which OPT finished within a set time limit. The results of MW, SW, and LP (with subsequent node promotion), as well as NS are close to each other with MW resulting in slightly narrower layerings. This conforms to the results of Nikolov et al. [11]. Concerning the actual minimum width, there is a gap of about 22 % between MW and OPT.

The results including OPT2 can be seen in Figure 3.5b, finishing for 1902 graphs.



(a) 3519 graphs (without OPT2)



(b) 1902 graphs (with OPT2)

Figure 3.5: The estimated widths (maximum number of original nodes and dummy nodes in any layer) for the different layering methods. OPT refers to the minimal possible estimated width, OPT2 refers to the minimal possible estimated width if the requirement is dropped that edges have to point into a common direction. OPT and OPT2 were implemented as optimization problems which did not finish for all graphs within a set time limit, thus only subsets of the graphs are plotted here. The boxplots and the line plots contain the same data.

This time there is a gap of about 75 % between the results of MW and OPT2. It must be noted however that MW is not allowed to let edges point upwards. Nevertheless, it gives a feeling of what is theoretically possible when seeking for layerings with a small width. Further note that it is possible that other metrics such as the number of dummy nodes, i. e. the edge length, and the layering's height may be significantly worse for OPT and OPT2. We did not investigate this so far.

3.2.1 Optimization Problem

We started our experiments with a variation of a straightforward model for the minimum-width layering problem presented by Healy and Nikolov [6], where we minimized the width instead of the number of dummy nodes. However, we were able to get a larger number of optimal results using an approach that combines *Constraint Programming (CP)* with *SAT solving* [12, 4]. Our new model is defined in the *MiniZinc language*[2], a high-level language for specifying optimization problems independently from a solver. We executed it using the open-source solver *chuffed*[3].

Inputs. Let $G = (V, E)$ be a graph with a set of nodes V and a set of edges E. Let $\{1, \ldots, n\}$ denote the nodes, where $n = |V|$.

Parameters. The layer of each node $i \in V$ is stored in a variable x_i and can take values in $\{1, \ldots, n\}$, where n is a trivial upper bound on the number of layers. For a layer $1 \leq l \leq n$, the variable l_l holds the number of regular nodes in this layer. Further, the variable w_l holds the sum of regular nodes as well as dummy nodes in layer l. $b2i$ is an operator that converts a boolean expression to an integer: true to 1 and false to 0.

Objective.

$$\text{Minimize} \qquad \max_{1 \leq l \leq n} w_l \tag{A}$$

Constraints.

$$x_i < x_j \qquad \text{for all } (i,j) \in E \tag{B}$$

$$l_l = \sum_{i \in V} b2i(x_i = l) \qquad \text{for all } 1 \leq l \leq n \tag{C}$$

$$w_l = l_l + \sum_{(i,j) \in E} b2i(x_i < l \wedge x_j > l) \qquad \text{for all } 1 \leq l \leq n \tag{D}$$

The constraints can easily be altered to allow edges with an arbitrary direction. For this, constraints (B) and (D) are replaced by the following two constraints.

$$x_i \neq x_j \qquad \text{for all } (i,j) \in E \tag{E}$$

$$w_l = l_l + \sum_{(i,j) \in E} b2i(\min\{x_i, x_j\} < l \wedge \max\{x_i, x_j\} > l) \qquad \text{for all } 1 \leq l \leq n \tag{F}$$

[2] `http://www.minizinc.org/`
[3] `http://github.com/geoffchu/chuffed`

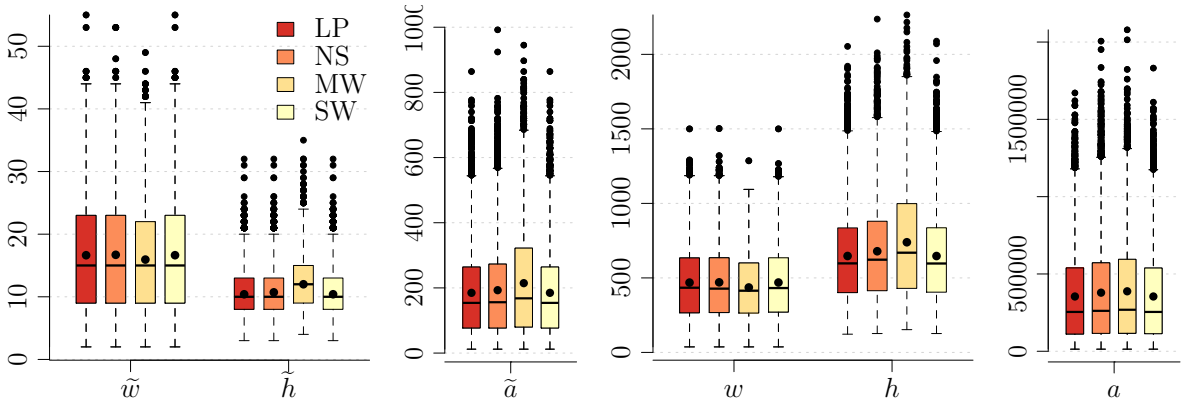


Figure 3.6: The width, height, and area of the RomeF graphs when drawn with different layering methods. Estimated measures (left side) are marked with a tilde.

3.3 Assessing Final Drawings

In the previous two sections estimated values were used during evaluations, primarily the estimated width of a layering. As mentioned before, we believe that it is not enough to assess the performance of a layering heuristic using estimations of the final dimensions of a drawing when it comes to practical applications. The layering phase only defines parts of the topology of a drawing and it is not until the node placement and edge routing phase that explicit coordinates are determined. For this reason, we extended the evaluation of Nikolov et al. by further measures, such as the effective width in pixels.

We created final drawings of the RomeF graphs using a node placement technique presented by Gansner et al. [5] and a simple polyline edge routing. The prescribed spacing between pairs of nodes is set to 20 pixels, both vertically and horizontally. The parameters *ubw* and *ubc* of MW were set to 4 and 2. We plotted the results in the same way as seen in Figure 3.1, i. e. a certain measure against the node count.

The only significant difference we could find is that without node promotion the estimated width of layerings created with MW is slightly larger than if the layering is created with NS, while the opposite is true for the effective width. Apart from that the overall trend of estimated and effective measures is comparable. We believe that the results are similar due to nodes being of the same size in the RomeF set and expect it to be different when nodes vary in size. Nevertheless, summarizing boxplots of our results can be seen in Figure 3.6. As said, the results of the four layering algorithms are similar for the presented measures with MW being a minor exception: the width of layerings created with MW is slightly smaller but the height of the layerings is larger. This is true for both estimated and effective values. Looking at the area, the reduction in width cannot compensate for the increase in height, resulting in an overall larger area for MW. Furthermore, while the estimated width is on average larger than the estimated height for all layering algorithms, it is the opposite way around for the effective width and effective height. A possible explanation for this could be different spacing values between pairs of layers and between pairs of nodes within the same layer. This is not the

case here since we set both spacings to the same value as mentioned above. It is more likely that the additional height is contributed by the final edge routing phase, which makes enough room between pairs of layers to tidily route the edges. Consequently, the estimated values obtained after the layering phase cannot be used as a reliable indicator for the aspect ratio of the final drawings.

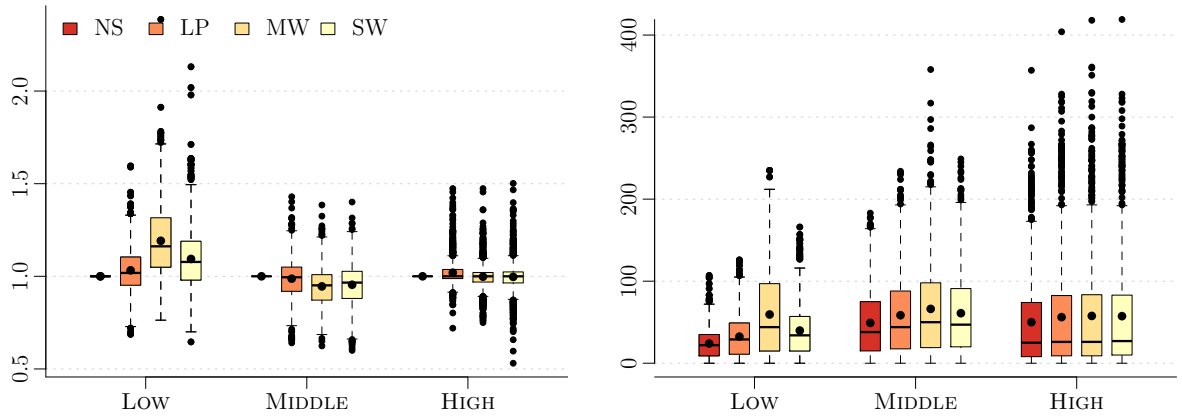
Summarizing this section, we found that while the estimations of width and height are similar to the width and height of a final drawing for the RomeF graphs, care has to be taken when drawing conclusions based on combinations of the measures, e. g. the aspect ratio. Moreover, MW indeed slightly reduces the width of a final drawing, however at the cost of an overall larger area. Generally speaking, no significant differences can be observed between the layering algorithms, particularly no advantages in terms of drawing area. This leads to the question whether MW and SW are useful in practice. We address this question in the next section; after all the idea to reduce the width while increasing the height sounds promising, for instance, in order to alter the layering of a graph such that it better suits a particular drawing area.

3.3.1 Target Drawing Area

Usually the drawing of a graph ends up being displayed on a screen or printed to a page. In both cases it is not enough to look at the width and height of a drawing in isolation to evaluate its quality. The max scale measure introduced in Chapter 2 grasps this desire to assess the quality of a drawing w.r.t. to a particular reference frame.

The previous section posed the question as to whether MW and SW are usable in practice. If one would plot max scale values for the layering methods discussed here and the RomeF graphs in the same way as in Figure 3.6, the conclusion would be that NS works best. Nevertheless, even if not usable as general purpose layering algorithms, MW and SW may be able to create layerings that are *more suitable* for a certain drawing area where NS fails. To evaluate this, we split the RomeF graphs into three further subsets, as explained in Section 2.2: Low, Middle, and High. MW and SW try to create layerings with a small width, thus we expect it to perform well for graphs for which NS uses few layers and places many nodes within the layers.

As a basis for the evaluation we chose the reference frame $\mathcal{R}_{(16,10)}$, resembling an up-to-date computer screen, and a left-to-right layout direction. The results seen in Figure 3.7 confirm our expectations on the one hand and indicate that one has to be careful when to apply a certain method on the other hand. The boxplots in (a) show max scale ratio values for MW, SW, and LP relative to NS's results. The boxplots in (b) show the number of dummy nodes for each method. For the Low graphs MW produces better results than NS for over three quarters of the graphs. However, a larger number of dummy nodes (thus edge length) is necessary which may worsen readability. The picture is completely different for Middle, where MW produces worse results. NS guarantees a minimum number of dummy nodes, and thus produces very compact drawings. Because the graphs of Middle were selected to roughly fit $\mathcal{R}_{(16,10)}$ when drawn with NS, it feels natural that MW performs worse. For High no conclusion can be drawn from the results. Still, since the graphs in this set are already quite narrow MW and SW would have to



(a) Max scale ratio

(b) Number of dummy nodes

Figure 3.7: Boxplots for the subsets of the RomeF graphs. A reference frame $\mathcal{R}_{(16,10)}$ and a left-to-right layout direction were used. Max scale ratio values are relative to the max scale values of NS.

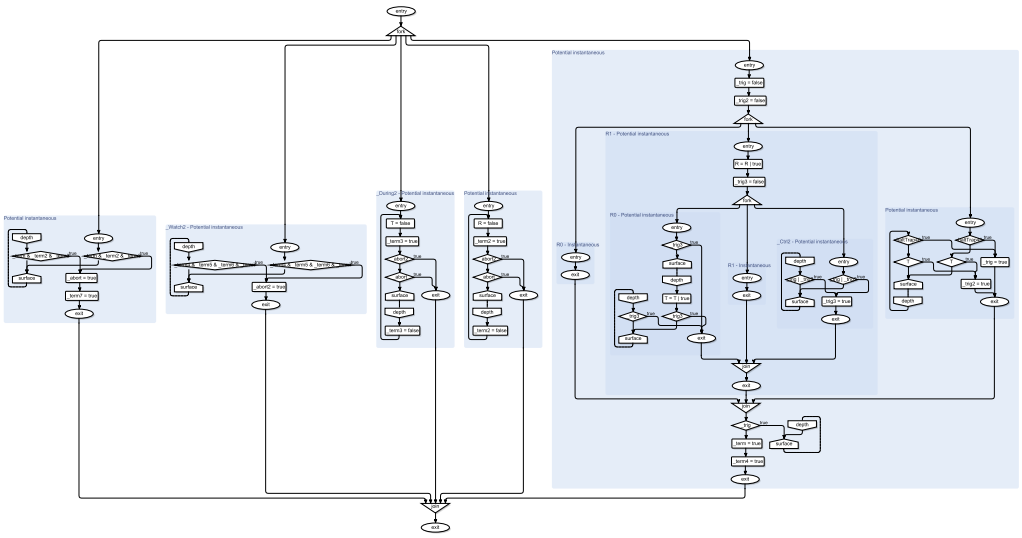
behave in the opposite way (putting more nodes in layers) in order to improve the max scale measure for $\mathcal{R}_{(16,10)}$.

To conclude, it makes sense to use MW and SW in cases where NS is not able to produce drawings that feature the desired aspect ratio.

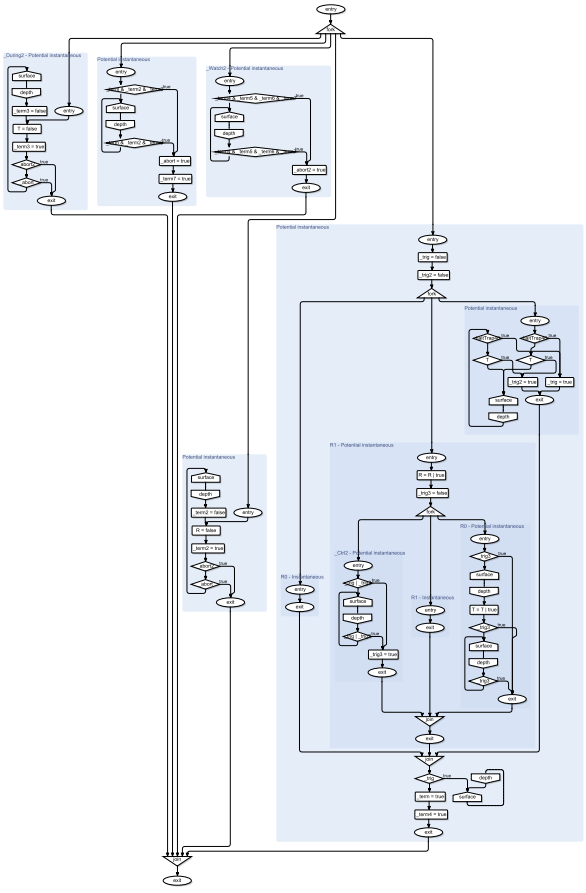
3.4 Considering Actual Node Sizes

During the previous sections regular nodes and dummy nodes were considered to have the same dimensions. In practice however, the dimension of the nodes of a graph may vary significantly depending on the application the graph originates from. Furthermore, the introduced dummy nodes effectively represent edges, which usually require less space in the final drawing than regular nodes. An example of this can be seen in Figure 3.8, which additionally contains *hierarchical nodes*, nodes that contain further nodes. To handle the hierarchy, the layout algorithm is executed in a bottom-up fashion, starting with the inner-most hierarchical nodes. Nikolov et al. already consider different node widths during their motivation of the problem but return to a unit width as soon as they discuss the heuristics. The remainder of this section discusses how actual node widths can be incorporated into each of the three heuristics. Note that it is essential that the node promotion has the same, or at least a very similar, notion of a layer's width as the prior layering heuristic. Otherwise it is not possible to prevent nodes from being promoted that would inadvertently increase the width of the layering.

To incorporate varying node dimensions we proceed in a similar fashion for MW and SW, and thus start with a general explanation before we discuss details specific to each method. The two heuristics essentially use the out-degree and the in-degree of a node v as predictions for the number of introduced dummy nodes, i. e. the contribution to the width of layers adjacent to v's layer. Thereby, a dummy node contributes one unit to



(a) Traditional methods



(b) MinWidth and PromoteNodes

Figure 3.8: Example of an SCG drawn with different layering methods. Depending on the available drawing area, one can imagine that either (a) or (b) is advantageous. For instance, (b) can be scaled larger when fitted on a sheet of paper in portrait orientation. Node labels would thus be more legible.

the width of a layer just as a regular node contributes one unit.

Let $w(v)$ denote the width of a node v and $\check{w}$ denote the smallest width of any node of a graph. During a pre-processing step, we normalized the node widths with respect to $\check{w}$ and refer to the normalized widths as $\bar{w}(v)$. Furthermore, we set the width of dummy nodes w^d to a user-specified minimum separation between edges and normalize it as well. Wherever necessary, one can have individual widths for dummy nodes, e. g. if edges have different line widths. We omit this during the upcoming explanations since it is easy to incorporate. Let w_c denote the width of the currently constructed layer and w_u denote the predicted width of future layers. With these definitions the decision whether to start a new layer can be adjusted for both MW and SW in the way discussed below. Additionally, the parts of the original heuristics that keep track of the current layer's width w_c and the predicted width of future layers w_u must multiply the node degrees by w^d and use the real width of a node. This can however be amended straightforwardly. Apart from these two modifications, the way the next node to be placed is selected remains unaltered.

3.4.1 MinWidth

Let ubw and ubc be the two input parameters to MW as discussed in Chapter 3. Further let v be the node currently looked at.

Size-Un-Aware. The original condition to start a new layer as defined by Nikolov et al. is:

$$w_c \geq ubw \ \wedge \ d^+(v) < 1$$

$$\text{or} \qquad w_u \geq ubw \cdot ubc$$

Size-Aware. We change this to consider actual node widths as follows. The upper bound on the width of a layer ubw is multiplied by the average width of all nodes of the graph. The resulting value is denoted by ubw'. If a node's width exceeds ubw' it can still be placed since MW first assigns nodes to the current layer and then checks the condition.

$$w_c \geq ubw' \ \wedge \ d^+(v) \cdot w^d < \bar{w}(v)$$

$$\text{or} \qquad w_u \geq ubw' \cdot ubc$$

3.4.2 StretchWidth

Let w_m denote the current maximal allowed width of a layer that is iteratively increased if it is too small (cf. Chapter 3). Further let v be the node currently looked at.

Size-Un-Aware. The original condition to start a new layer as defined by Nikolov et al. is:

$$w_c - d^+(v) + 1 \;>\; w_m$$

$$\text{or} \qquad w_u + d^-(v) \;>\; w_m \cdot \bar{d}^+(G)$$

Size-Aware. We change this to consider actual node widths as follows:

$$w_c - d^+(v) \cdot w^d + \bar{w}(v) \;>\; w_m$$

$$\text{or} \qquad w_u + d^-(v) \cdot w^d \;>\; w_m \cdot \bar{d}^+(G) \cdot w^d$$

In the original version of the algorithm the w_u variable only accumulates the in-degrees of the nodes placed within the current layer and is reset every time a new layer is started. As a consequence long edges spanning multiple layers are only considered once and neglected afterwards, i. e. they only contribute width to one layer no matter how long they are. An alternative could be to not reset the w_u variable when a new layer is started but to "correct" it instead: once a node v is placed in the current layer, v's out-degree is subtracted from w_u and its in-degree is added.

We briefly evaluated this for the SCGs, where we observed that the second part of the condition to start a new layer is hardly true as w_u is often too small. Our modification seemed to improve things slightly but requires more thorough evaluation to make a final statement.

3.4.3 PromoteNodes

The PN heuristic tries to improve existing layerings by reducing the overall number of dummy nodes. When applied after either MW or SW, which try to create layerings with small width, it is sensible to prevent PN from increasing the layering's width again.

As soon as MW and SW are aware of width of regular nodes and dummy nodes, PN must consider this as well. It is not hard to incorporate this into the heuristic. The width of every layer of the given layering can be computed upfront and it makes no differences whether regular nodes and dummy nodes factor in with a width of one or an individual width. While promoting nodes it can be checked if moving a node would increase the maximal width of the original layering.

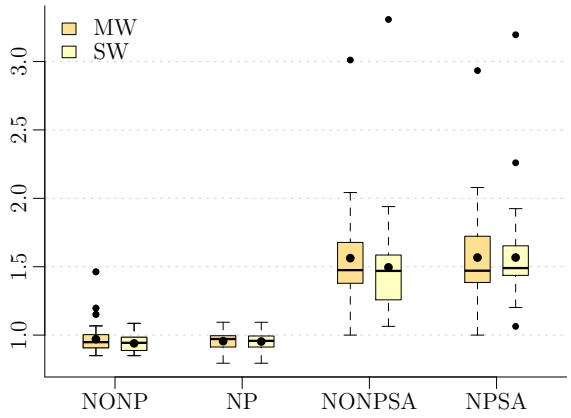


Figure 3.9: Assessing the impact of considering individual node sizes based on the set of SCGs A reference frame $\mathcal{R}_{(10,16)}$ is assumed. The plotted max scale ratio values are relative to NS's results.

3.4.4 Results

The example in Figure 3.8 highlights the importance of considering actual node sizes. Note that the depicted graph is hierarchical. It is laid out in a bottom-up fashion, consecutively executing the layout algorithm for the different hierarchy levels. As such, the shaded areas representing hierarchical nodes are at some point "black boxes" from the perspective of the layout algorithm.

Looking at the top-level graph with 9 nodes and 12 edges, the estimated width of the layering in both (a) and (b) is 5 when individual node widths are not considered. In their original form, neither MW nor SW would create the drawing (b), which in terms of pixels is significantly narrower.

We executed the two layering heuristics for the set of SCGs, as introduced in Section 2.2, in four configurations: either without node promotion (NONP) or with node promotion (NP), and either without size-awareness or with size-awareness (SA). Remember that SCGs are drawn top-down and that we selected those test graphs that are rather wide (large aspect ratio). This time we aim at producing drawings that fit computer screens in portrait-orientation, i. e. $\mathcal{R}_{(10,16)}$. The results are presented in Figure 3.9. The boxplot shows max scale ratio values relative to the results produced with the network simplex layering (NS) approach. The size-aware heuristics clearly produce better drawings for the defined setting when compared to the original versions, with drawings that can be displayed about 1.5 times larger. When not considering node sizes the results of the heuristics are worse than NS for the majority of the graphs. No clear winner can be identified between MW and SW based on the tested SCGs. The execution of a subsequent node promotion only improves matters for SW. However, the latter two observations probably strongly depend on the particular set of graphs.

4 Discussion

In this report we examined layering heuristics for the minimum-width layering problem and posed three questions in Chapter 1. Summarizing they can be answered as follows.

First, we found that for many of the RomeF graphs there is a noteworthy gap between the heuristic solution and an optimal solution. Nevertheless, it is not clear if optimal solutions are desirable when it comes to a layering's width. After all, a very narrow drawing may corrupt other important layout aesthetics, such as the number of edge crossings and the overall edge length. Second, in the general case the layering heuristics perform inferior to the traditional layering method NS. However, when used with care MW and SW are able to produce drawings that are better suited for certain drawing areas, e. g. computer screens. Third, we showed how one can extend the heuristics to consider individual node sizes. Our evaluation based on a set of SCGs suggests that this is both necessary and successful.

Finally, we look at future avenues of research. Both heuristics are based on the idea of the LP algorithm. Superfluous dummy nodes are removed by the node promotion post-processing step. To a certain extent this mimics the underlying goal of NS: minimizing the number of dummy nodes. It has been shown that NS produces good and compact layerings [7]. Therefore, it may be more promising to base a heuristic for the minimum-width layering problem on the idea of NS from the start. Another point to consider is whether MW's parameters should be linked to the graph instance at hand, instead of being absolute values. The parameter values suggested by Nikolov et al. base on extensive experiments with the RomeF graphs [11]. Those graphs have a specific range of node and edge counts and are sparse. Denser graphs may require larger parameter values for good results.

Acknowledgments. We thank Thorsten Ehlers for valuable support regarding constraint programming and the use of chuffed.

Bibliography

[1] Jürgen Branke, Stefan Leppert, Martin Middendorf, and Peter Eades. Width-restricted layering of acyclic digraphs with consideration of dummy nodes. *Information Processing Letters*, 81:59–63, 2002.

[2] Edward G. Coffman. and Ronald L. Graham. Optimal scheduling for two-processor systems. *Acta Informatica*, 1(3):200–213, 1972.

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

[4] Thibaut Feydy and Peter J. Stuckey. Lazy clause generation reengineered. In Ian P. Gent, editor, *Principles and Practice of Constraint Programming - CP 2009, 15th International Conference, CP 2009, Lisbon, Portugal, September 20-24, 2009, Proceedings*, volume 5732 of *Lecture Notes in Computer Science*, pages 352–366. Springer, 2009.

[5] Emden R. Gansner, Eleftherios Koutsofios, Stephen C. North, and Kiem-Phong Vo. A technique for drawing directed graphs. *Software Engineering*, 19(3):214–230, 1993.

[6] Patrick Healy and Nikola S. Nikolov. A branch-and-cut approach to the directed acyclic graph layering problem. In Stephen G. Kobourov and Michael T. Goodrich, editors, *Proceedings of the 10th International Symposium on Graph Drawing (GD'02)*, volume 2528 of *LNCS*, pages 98–109. Springer, 2002.

[7] Patrick Healy and Nikola S. Nikolov. How to layer a directed acyclic graph. In Petra Mutzel, Michael Jünger, and Sebastian Leipert, editors, *Proceedings of the 9th International Symposium on Graph Drawing (GD'01)*, volume 2265 of *LNCS*, pages 16–30. Springer, 2002.

[8] Patrick Healy and Nikola S. Nikolov. Characterization of layered graphs with the minimum number of dummy vertices, 2003.

[9] Patrick Healy and Nikola S. Nikolov. Hierarchical drawing algorithms. In Roberto Tamassia, editor, *Handbook of Graph Drawing and Visualization*, pages 409–453. CRC Press, 2013.

[10] Nikola S. Nikolov and Alexandre Tarassov. Graph layering by promotion of nodes. *Discrete Applied Mathematics*, 154(5):848–860, 2006.

[11] Nikola S. Nikolov, Alexandre Tarassov, and Jürgen Branke. In search for efficient heuristics for minimum-width graph layering with consideration of dummy nodes. *Journal of Experimental Algorithmics*, 10, 2005.

[12] Olga Ohrimenko, Peter J. Stuckey, and Michael Codish. Propagation = lazy clause generation. In Christian Bessiere, editor, *Principles and Practice of Constraint Programming – CP 2007, 13th International Conference, CP 2007, Providence, RI, USA, September 23-27, 2007, Proceedings*, volume 4741 of *Lecture Notes in Computer Science*, pages 544–558. Springer, 2007.

[13] Kozo Sugiyama, Shojiro Tagawa, and Mitsuhiko Toda. Methods for visual understanding of hierarchical system structures. *IEEE Transactions on Systems, Man and Cybernetics*, 11(2):109–125, February 1981.

[14] Alexandre Tarassov, Nikola S. Nikolov, and Jürgen Branke. A heuristic for minimum-width graph layering with consideration of dummy nodes. *Experimental and Efficient Algorithms*, pages 570–583, 2004.

[15] Reinhard von Hanxleden, Michael Mendler, Joaquín Aguado, Björn Duderstadt, Insa Fuhrmann, Christian Motika, Stephen Mercer, Owen O'Brien, and Partha Roop. Sequentially Constructive Concurrency—A conservative extension of the synchronous model of computation. *ACM Transactions on Embedded Computing Systems, Special Issue on Applications of Concurrency to System Design*, 13(4s):144:1–144:26, July 2014.