INSTITUT FÜR INFORMATIK

A Generalization of the Directed Graph Layering Problem

Ulf Rüegg, Thorsten Ehlers,
Miro Spönemann, and Reinhard von Hanxleden

Bericht Nr. 1501
February 2015
ISSN 2192-6247

CHRISTIAN-ALBRECHTS-UNIVERSITÄT

ZU KIEL

Institut für Informatik der
Christian-Albrechts-Universität zu Kiel
Olshausenstr. 40
D – 24098 Kiel

A Generalization of the Directed Graph Layering Problem

Ulf Rüegg, Thorsten Ehlers,
Miro Spönemann, and Reinhard von Hanxleden

Bericht Nr. 1501
February 2015
ISSN 2192-6247

U. Rüegg, T. Ehlers, and R. von Hanxleden are with the
Department of Computer Science, Kiel University, Kiel, Germany.
E-mail: {uru,the,msp,rvh}@informatik.uni-kiel.de

M. Spönemann is with the itemis AG, Kiel, Germany
E-mail: miro.spoenemann@itemis.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 Directed Layering Problem (DLP) solves a step of the widely used layer-based layout approach to automatically draw directed acyclic graphs. To cater for cyclic graphs, classically 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 show GLP to be NP-complete, present integer programming models to solve it, 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.

Keywords: layer-based layout, Sugiyama layout, linear arrangement, feedback arc set, bandwidth sum coloring, integer programming, data flow diagrams

Contents

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 subsequent *layers* clearly showing the inherent direction of the graph, see Figure 1.1 for an example. The approach was introduced by Sugiyama et al. [21] and remains a subject to 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*:* Remove 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 crossings is minimized.

4. *Coordinate assignment:* Determine explicit node coordinates with the goal to draw as many edges straight as possible.

5. *Edge routing*:* Route the edges depending on the application, e. g. using an orthogonal style.

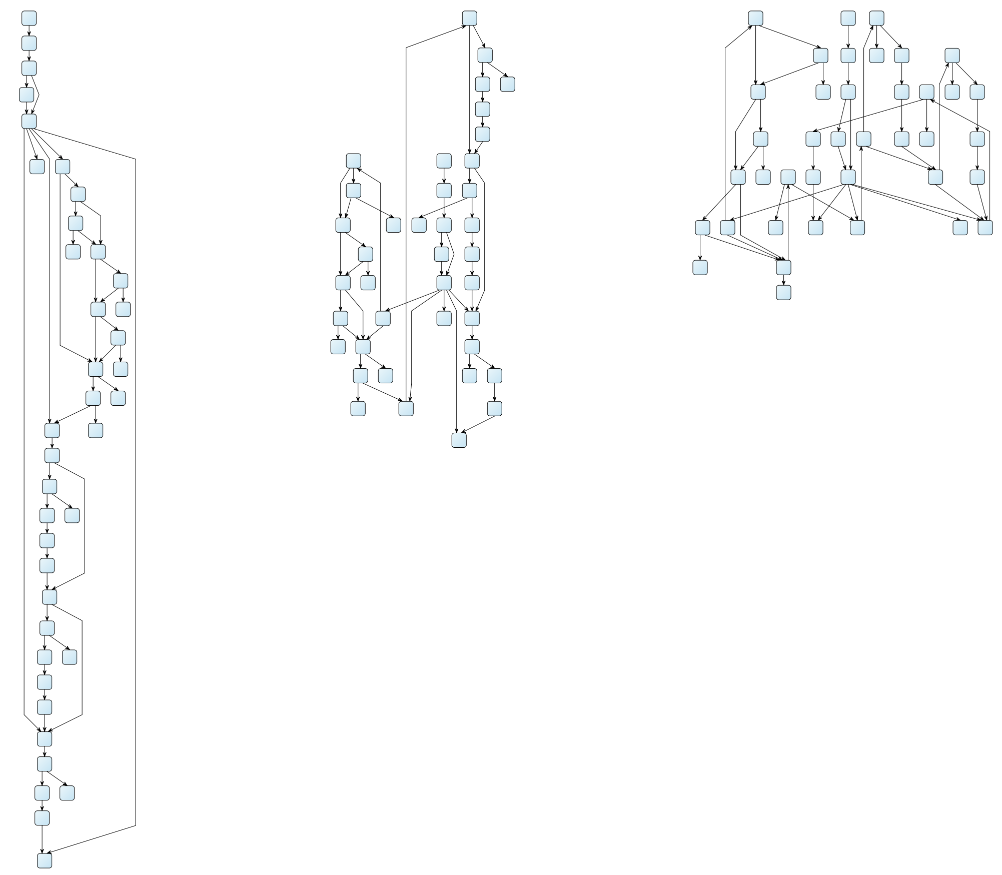
While state-of-the-art methods produce drawings that are often satisfying, there are graph instances where the results show bad *compactness* and unfavorable *aspect ratio* [11]. In particular, the number of layers is bound by the longest path of the input graph after the first phase. When placing the layers vertically one above the other this affects the height of the drawing, see for instance Figure 1.1a.

Following these observations, we present new methods to overcome current limitations.

1.1 Contributions

The focus of this paper is on the first two phases, as they have the greatest impact on the compactness and the aspect ratio of the resulting drawing. Our main new contributions are the following.

We introduce a new layer assignment method named *GLay*, which is able to handle cyclic graphs and to consider compactness properties for selecting an edge reversal set.



(a) 0 reversed edges, 71 dummy nodes

(b) 2 reversed edges, 35 dummy nodes

(c) 6 reversed edges, 16 dummy nodes

Figure 1.1. Different drawings of the g.39.29 graph from the AT&T graphs collection. (a) is drawn with known methods described in Chapter 4. (b) and (c) are results of the methods presented here.

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 the number of reversed edges for certain graphs. See Figure 1.1 and Figure 4.2 for examples.

We present and discuss integer programming models to solve the new problem to optimality, which we show to be NP-complete. The models can be extended to incorporate hard bounds on the number of layers and maximum number of nodes in any layer.

We perform thorough evaluations of the new approach using graphs from practical applications as well as randomly generated graphs. The results show that the new method easily improves drawings of graphs for which current methods, as described in Chapter 4, produce drawings with unfavorable aspect ratio. They furthermore show an improved average performance in terms of compactness for general graphs without explicitly aiming for compactness.

Outline. The next section presents related work. We introduce problems and definitions in Chapter 2 and present models to solve the newly introduced problems in Chapter 3. Chapter 4 discusses thorough evaluations before we conclude in Chapter 5.

1.2 Related Work

The cycle removal phase targets the Feedback Arc Set Problem (FASP) which is defined more precisely in the next section. This problem is NP-complete and several approaches have been proposed to solve the problem either to optimality or heuristically. The most popular heuristic used in the context of graph drawing was introduced by Eades et al. [7]; it yields reasonable results and runs in linear time.

It has been noted before, however, that reversing a minimal number of edges does not necessarily yield the best suiting results [9] and that application-inherent information might make certain edges better candidates to be reversed. 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 (cf. Figure 1.2), nevertheless the two phases are executed separately until today.

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

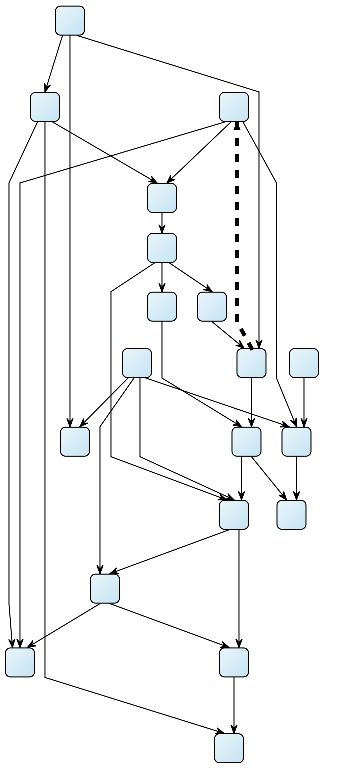
Eades and Sugyiama employ a longest path layering which requires linear time and the resulting number of layers equals the number of nodes of the graph's longest path [8].

Gansner et al. solve the layering phase by minimizing the sum of the edge lengths of the graph's edges [9]. 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, however, in practice it runs fast. This approach was found to inherently produce compact drawings and performed best in comparison to other layering approaches [13].

In a series of papers, 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 [13]. They show the problem to be NP-hard, even without considering dummy nodes, and evaluate the approach using linear programming. In a subsequent paper they present a branch-and-cut algorithm to solve the problem faster and for larger graph instances [12]. Later, Nikolov et al. [18, 17] propose and evaluate several heuristics to find a layering with a restricted number of nodes in each layer. Furthermore, an approach using ant colony optimization is presented by Andreev et al. [1].

Nachmanson et al. present an iterative algorithm to produce drawings with an aspect ratio close to a previously specified value [16].

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 of the graph. This particularly means that the bound on the number of layers in the methods of Nikolov et al. cannot be smaller than the longest path.


(a) EaGa – 46 dummy nodes

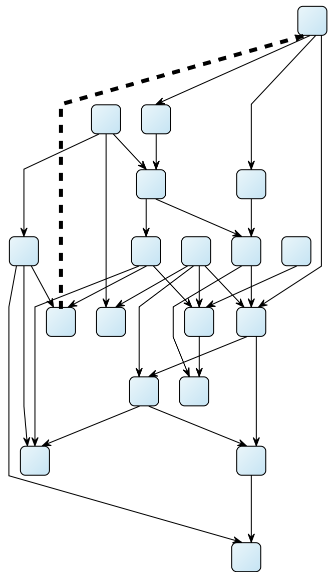

(b) OptGa – 20 dummy nodes

Figure 1.2. Two drawings of the same graph with 19 nodes. While in both drawings exactly one edge is reversed, different edges (dashed lines) are reversed. The example highlights the impact of the choice of which edge to reverse on the resulting number of dummy nodes.

2 Problem Descriptions and Definitions

In this section we define the terms and problems that are relevant for the rest of this paper. New problems that we introduce are marked with an asterisk.

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 mapping $L : V \to \mathbb{N}$.

Definition 2.2. Let $G = (V, E)$ be a directed acyclic graph. A layering L is *valid* if $\forall (u, v) \in E$: $L(v) - L(u) \geq 1$. An edge $(u, v) \in E$ is *short* in a valid layering if $|L(v) - L(u)| = 1$, 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.

We now introduce several minimization problems that are either related to the methods we present here or are used for the NP-completeness proofs.

Problem (Feedback Arc Set (FASP)). Given a directed graph $G = (V, E)$, find a minimum k such that $E' \subseteq E$ is a subset containing at least one edge of every directed cycle in G and $|E'| = k$. Such a subset E' is called Feedback Arc Set (FAS). This problem is one of Karp's 21 NP-complete problems [14].

An alternative formulation is to find a minimum k and a one-to-one mapping $f : V \mapsto \{1, \ldots, |V|\}$ such that $|\{(u, v) \in E : f(u) > f(v)\}| = k$.

Problem (Linear Arrangement (LAP)). Let $G = (V, E)$ be a graph. The problem seeks a minimum k and a one-to-one mapping $f : V \mapsto \{1, \ldots, |V|\}$ such that $\sum_{\{u,v\} \in E} |f(u) - f(v)| = k$.

The problem is NP-complete (cf. [GT42] in [10]) and also known as the *Bandwidth Sum Problem* [6]. It has a range of applications, is subject to ongoing research and several heuristics have been proposed, see for instance Caprara et al. for a literature research [4].

Problem (Sum Coloring (SCP)). Let $G = (V, E)$ be a graph. Kubicka and Schwenk introduced the concept of Sum Coloring Problem (SCP) [15] which seeks a minimum k and a mapping $c : V \mapsto \mathbb{N}$ such that $\forall \{u, v\} \in E : c(u) \neq c(v)$ and $\sum_{v \in V} c(v) = k$.

The problem is NP-complete and a minimum k is also called *chromatic sum* of G, $\Sigma(G)$.

Problem* (**Bandwidth Sum Coloring (BSCP)**). Let $G = (V, E)$ be a graph. Similar to the previous problems, we define the Bandwidth Sum Coloring Problem (BSCP) as finding a minimum k and a mapping $c : V \mapsto \mathbb{N}$ such that $\forall (u, v) \in E : c(u) \neq c(v)$ and $\sum_{(u,v)\in E} |c(u) - c(v)| = k$.

We prove this problem to be NP-complete in Section 2.1.

Problem* (**0/1-BSCP**). Let $G = (V, E)$ be a graph, we extend BSCP by a weight $w_e \in \{0, 1\}$ for every edge $e \in E$. The problem then seeks for a minimum k and a mapping $c : V \mapsto \mathbb{N}$ such that $\forall \{u, v\} \in E : c(u) \neq c(v)$ and $\sum_{\{u,v\}\in E} w_{\{u,v\}} |c(u) - c(v)| = k$.

We later use this problem to prove the NP-completeness of BSCP.

The following problems are the basis for the layering methods presented by Gansner et al. and for the course of this paper.

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

The problem was originally introduced by Gansner et al. to solve the layering problem for layer-based layout [9].

We now extend the idea of a layering for directed acyclic graphs to general graphs, i. e. graphs that are either directed or undirected and can possibly be cyclic.

Definition 2.3. Extending Definition 2.2, a layering L of a graph $G = (V, E)$ is *feasible* if $\forall (u, v) \in E : |L(u) - L(v)| \geq 1$. An edge $(u, v) \in E$ is *short* in a feasible layering if $|L(v) - L(u)| = 1$, otherwise it is *long*. A feasible layering L is *proper* if all edges are short.

Problem* (**Generalized Layering (GLP)**). We extend the DLP and define the Generalized Layering Problem (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 of G, i. e. a mapping L, such that $\forall (v, w) \in E : |L(w) - L(v)| \geq 1$ and

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

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 in the sense of Definition 2.2. 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 comparable to DLP for acyclic input graphs.

Problem* (ϵ-GLP). We define ϵ-GLP as a variation of GLP where we fix the size of the FAS instead of minimizing it. Let $G = (V, E)$ be a graph and s be an integer. The problem is to find a minimum k and a feasible layering, i. e. a mapping L such that $\forall (v, w) \in E : |L(w) - L(v)| \geq 1$, $|\{(v, w) \in E : L(v) > L(w)\}| = s$, and $\sum_{(v,w)\in E} |L(w) - L(v)| = k$.

The remainder of this section defines terms associated with the characteristics used to qualify a drawing of a graph.

Definition 2.4. Let G be a graph with a layering L. The *height* of the layering is the number of layers. The *width* is the maximum number of nodes in a layer over all layers. The width can optionally include dummy nodes.

Definition 2.5. Let G be a graph with a layering L. The *estimated area* is the width of L multiplied by its height. The *estimated aspect ratio* is the quotient of width and height.

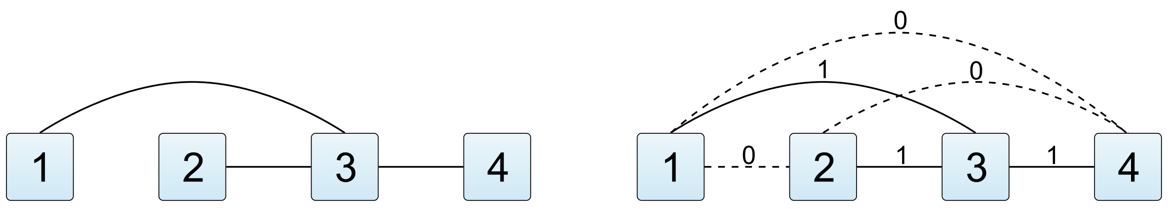
Definition 2.6. Let G be a graph with a layering L. The *edge density* of a layer i is the number of edges spanning from layer i to layer $i + 1$. The edge density of the last layer is thus always 0.

Definition 2.7. Let G be a graph. A *drawing* of G is an embedding in the plane.

Definition 2.8. Let D be a drawing of a graph G. The *width* of D is the difference between the maximal and the minimal horizontal component of any point in the embedding D. The *height* is defined symmetrically for the vertical components.

Definition 2.9. Let D be a drawing of a graph G. The *area* is the product of the width and height of D. The *aspect ratio* is the quotient of width and height.

Here we define desired aspect ratios to be close to the aspect ratios of paper sheets ($\sqrt{2} \approx 1.4$) or computer displays (between 1.3 and 1.8). While in the literature of the layer-based approach the usual edge direction is top-to-bottom, there are applications, e. g. data flow diagram layout, that are by convention drawn from left-to-right. For the rest of this paper we conform to the usual assumption of top-to-bottom layout and also conduct our experiments in this way. Note, however, that with a top-to-bottom drawing direction the desired aspect ratios for data flow diagram layouts are therefore in the range between 0.55 and 0.75. The resulting drawing is then rotated by 90° for presentation.



(a) Example input G for LAP **(b)** Constructed G' for 0/1-BSCP

Figure 2.1. Illustration of the construction of G' for Lemma 2. Edges with weight 0 are added to G until G' represents a complete graph.

2.1 NP-Completeness of BSCP and GLP

In the following, we examine the complexity of BSCP and GLP. For this we first prove an auxiliary lemma and show 0/1-BSCP to be NP-complete.

Lemma 1. Let $G = (V, E)$ denote a graph with n nodes. If there exists a feasible solution for 0/1-BSCP, then there exists one using at most n consecutive integers.

Proof. Let c be a mapping for 0/1-BSCP that yields a feasible solution. We may assume that $\min_{v \in V} c(v) = 1$ by a shifting argument. Now assume that $m_V = \max_{v \in V} c(v) > n$. That is, there is at least one integer $\ell \in 1, \ldots, m_V$ that no node is assigned to.

Let $V = V_1 \cup V_2$ be a partition of the nodes such that $V_1 \cap V_2 = \emptyset$, $c(v) < \ell$ for all $v \in V_1$, and $c(v) > \ell$ for $v \in V_2$. Let $c'(v) = c(v)$ for $v \in V_1$, and $c'(v) = c(v) - 1$ for $v \in V_2$. This yields $|c'(u) - c'(v)| = |c(u) - c(v)|$ if u and v are in the same partition, and $|c'(u) - c'(v)| = |c(u) - c(v)| - 1$ otherwise. Thus,

$$\sum_{e=\{u,v\} \in E} w_e |c(v) - c(u)| \geq \sum_{e=\{u,v\} \in E} w_e |c'(v) - c'(u)|$$

By induction, this concludes the proof. □

Lemma 2. 0/1-BSCP is NP-complete.

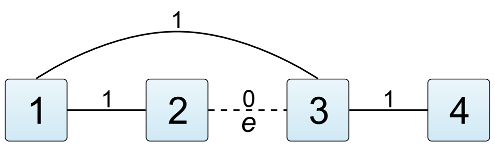
Proof. 0/1-BSCP obviously is in NP. For this, randomly assign integers to the nodes and verify (in polynomial time) that the edge sum is as required.

We now reduce LAP to 0/1-BSCP. For a set S let $\binom{S}{2}$ denote the set of all 2-subsets of S, i. e. $\binom{S}{2} = \{\{x, y\} : x, y \in S\}$.

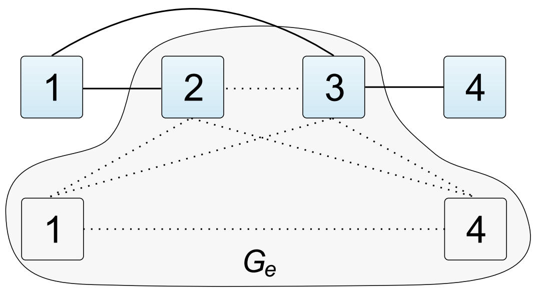
Let $G = (V, E)$ denote a graph on n nodes. Let $G' = (V, E')$, $E' = \binom{V}{2}$ denote the complete graph on n nodes, and let $w_e = 1$ iff $e \in E$ and $w_e = 0$ iff $e \in E' \setminus E$ (see Figure 2.1 for an illustration). We show that an optimum solution for 0/1-BSCP on G' is an optimum solution for LAP on G.

Let k_{LAP} and $k_{0/1-BSCP}$ denote minimum k for LAP and 0/1-BSCP, respectively. Moreover, let c and c' denote the mappings that produce k_{LAP} and $k_{0/1-BSCP}$.

As any solution for LAP is a feasible solution for 0/1-BSCP, we have $k_{LAP} \geq k_{0/1-BSCP}$. According to Lemma 1, there is an optimum solution for 0/1-BSCP using at most n



(a) Example input G for 0/1-BSCP



(b) Constructed G' for BSCP

Figure 2.2. Illustration of the construction of G' for Theorem 1. For all edges e of G with weight 0 auxiliary complete graphs G_e are added to G'. Node labels denote the integer value assigned to that node.

consecutive integers. Furthermore, by construction of G' we know that c' uses at least n different integers because G' is complete and for all $\{u, v\} \in E'$ it must hold that $c'(u) \neq c'(v)$.

Thus, exactly n consecutive integers must be used for G', and hence every solution for 0/1-BSCP is a solution for LAP as well. Thus, we have $k_{LAP} = k_{0/1-BSCP}$. $\square$

Theorem 1. BSCP is NP-complete.

Proof. BSCP obviously is in NP using the same argument as in Lemma 2.

We show that 0/1-BSCP can be solved using BSCP. Let $G = (V, E)$ denote a graph with n nodes and weights $w_e \in \{0, 1\}$ for edges $e \in E$. For all $e = \{u, v\} \in E$ where $w_e = 0$ let $G_e = (V_e, E_e)$, $V_e = \{u, v, x_{e_1}, \ldots, x_{e_{n-2}}\}$, $E_e = \binom{V_e}{2}$ denote a complete graph with $n - 2$ new nodes. We use these graphs to construct

$$G' = (V', E') = \left(V \cup \bigcup_{e \in E | w_e = 0} V_e,\ E \cup \bigcup_{e \in E | w_e = 0} E_e \right) .$$

Clearly, this construction is polynomial. An example can be seen in Figure 2.2.

Let $BSCP(G', k')$ denote that there is a solution for BSCP on G' with a given k', and 0/1-$BSCP(G, k)$ be equivalent for 0/1-BSCP on G and k. Furthermore, let $z = |\{e \in E : w_e = 0\}|$ denote the number of edges with weight 0. We show that

$$BSCP\left(G', k + z \cdot \binom{n+1}{3}\right) \Longleftrightarrow 0/1\text{-}BSCP(G, k) .$$

For the implication "$\Leftarrow$", let c denote a solution of 0/1-BSCP for G and k. By Lemma 1 we may assume that the nodes are assigned to at most n consecutive integers. Create the assignment c' as follows. Assign nodes from $V' \setminus V$ greedily to positive integers smaller than n such that for all $e \in E$ with $w_e = 0$ and all $v_1 \neq v_2 \in V_e : c'(v_1) \neq c'(v_2)$.

Additionally, retain the assignments for all $v \in V$. Let $E_1 = \{e \in E : w_e = 1\}$, and $E_2 = E' \setminus E_1$. Thus, we have

$$\begin{aligned}\sum_{\{u,v\} \in E'} |c'(u) - c'(v)| &= \sum_{\{u,v\} \in E_1} |c'(u) - c'(v)| + \sum_{\{u,v\} \in E_2} |c'(u) - c'(v)| \\ &= k + z \cdot \binom{n+1}{3} \ ,\end{aligned}$$

as the costs for a complete graph on n nodes are exactly $\sum_{1 \leq x < y \leq n} |x - y| = \binom{n+1}{3}$.

Next, for the implication "$\Rightarrow$", let us assume that we are given a solution c' for BSCP with $k' = k + z \cdot \binom{n+1}{3}$. The costs induced by the graphs G_e are lower-bounded by $z \cdot \binom{n+1}{3}$, thus there is an assignment for the remaining nodes with costs upper-bounded by k. $\square$

Theorem 2. GLP is NP-complete.

Proof. Obviously, GLP can be used to solve both FAS and BSCP, either by setting $\omega_{\text{rev}} = 0$ (BSCP) or $\omega_{\text{len}} = 0$ (FAS). $\square$

3 Solving the Generalized Layering Problem

In the following, we describe how to solve the GLP using integer programming (IP). As discussed above, the problem is NP-complete. An IP formulation allows us to conduct first experiments with optimal solutions to indicate the usefulness of the presented approach. We present three different IP formulations. One turns out to perform quite well in terms of execution time for medium-sized problems, while another version using only binary variables performs poorly. Third, we present an IP formulation for ϵ-GLP.

3.1 GLay IP Model

The rough idea of the following model is to assign an integer value to each node of the given graph that represents the layer in which the 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 $u, v \in V$.

Objective.

$$\begin{aligned} \text{Minimize} \quad & \omega_{\text{len}} \cdot \sum_{(u,v) \in E} |l(u) - l(v)| && \text{(A)} \\ & + \omega_{\text{rev}} \cdot \sum_{(u,v) \in E} r(u, v) && \text{(B)} \end{aligned}$$

Part A represents the edge lengths, i. e. the number of dummy nodes. Part B is the number of reverse edges.

Constraints.

1. $l(v) \geq 1 \quad \forall v \in V$
2. $l(v) \leq n \quad \forall v \in V$
3. $|l(u) - l(v)| \geq 1 \quad \forall (u,v) \in E$
4. $n \cdot r(u,v) + l(v) \geq l(u) + 1 \quad \forall (u,v) \in E$

Constraints (1) and (2) restrict the range of possible layers. (3) ensures that the resulting layering is feasible. (4) binds the decision variables in r to the layering and is a realization of $r(u,v) = 0 \Rightarrow L(v) > L(u)$. We can omit the reverse implication $r(u,v) = 1 \Rightarrow L(v) < L(u)$ if r is part of the objective function and $\omega_{\text{rev}} > 0$.

The model contains non-linear constraints due to the absolute values. We implemented the model in CPLEX where an absolute-value function is available off the shelf. For graphs with up to 50 nodes the execution on an up-to-date laptop takes between 2 and 15 seconds.

Variations. The model can easily be extended to restrict the number of layers by replacing the n in constraint (2) 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.

3.2 ϵ-GLay IP Model

ϵ-GLP is a variation of GLP where we remove the number of reversed edges from the objective function and instead fix it to a certain number s by adding it as a constraint. For this we also have to adapt constraint (4) as r is not part of the objective anymore. In fact, we get rid of r altogether and add another constraint as follows.

Objective.

$$\text{Minimize} \quad \sum_{(u,v) \in E} |l(u) - l(v)| \tag{C}$$

Constraints. Let $\text{neg}(x) = 1$ if $x < 0$ and $\text{neg}(x) = 0$ otherwise for all $x \in \mathbb{Z}$.

1. $l(v) \geq 1 \quad \forall v \in V$
2. $l(v) \leq n \quad \forall v \in V$
3. $|l(u) - l(v)| \geq 1 \quad \forall (u,v) \in E$
4. $\sum_{(u,v) \in E} \text{neg}(l(u) - l(v)) = s$

Constraints (1), (2), and (3) are defined as in Section 3.1. (4) assures that exactly s edges are reversed (in case s allows a feasible solution). Note that with CPLEX the $\text{neg}(x)$ operator can be directly implemented by writing $(x < 0)$.

Again, one can easily add a bound on the number of layers and define different weights for the edges.

3.3 GLay 0-1 IP Model

We also experimented with a model containing only boolean decision variables. The rough idea is to have one decision variable for each combination of node and layer that is set if the node is placed in that layer. This yields a number of decision variables equal to the number of nodes times the number of possible layers. For each edge two decision variables are introduced indicating whether the edge has a certain length and whether it is reversed.

However, the model performed poorly in terms of execution time and did not allow proper evaluations. We present it here for completeness since it can serve as a basis to incorporate further constraints. For example, contrary to the IP of the previous section, the following model is able to directly incorporate the notions of dummy nodes and number of nodes per layer.

Input and parameters.
Let $G = (V, E)$ be a graph with node set $V = \{1, \dots, n\}$. Let $m = |E|$ and $[m] = \{1, \dots, m\}$. ω_{len} and ω_{rev} are weighting constants.

Boolean decision variables.
$l(v, i) = 1$ iff node v is in layer $i \in [n]$.
$el(e, d) = 1$ iff edge e has length d, i. e. if for an edge $e = (u, v)$ with $l(u, i) = 1$, $l(v, j) = 1$, and $|i - j| = d$ then $el(e, d) = 1$.
$er(e, r) = 1$ for an edge $e = (u, v)$ with $r = j - i + n$, $l(u, i) = 1$, and $l(v, j) = 1$. We consider e to be reversed if $r \leq n$.

Objective.

$$\text{Minimize} \quad \omega_{\text{len}} \cdot \sum_{e \in [m], d \in V} d \cdot el(e, d) \tag{D}$$

$$+ \, \omega_{\text{rev}} \cdot \sum_{e \in [m], r \in V} er(e, r) \tag{E}$$

Part D represents the edge lengths, i. e. number of dummy nodes. Part E is the number of reversed edges.

Constraints.

1. $\sum_{x \in [n]} l(v,x) = 1 \quad \forall v \in V$
2. $l(u,x) + l(v,x) \leq 1 \quad \forall (u,v) \in E, x \in [n]$
3. $-l(u,i) - l(v,j) + el((u,v),d) \geq -1 \quad \forall (u,v) \in E, d = |i-j|$
4. $-l(u,i) - l(v,j) + er((u,v),r) \geq -1 \quad \forall (u,v) \in E, r = j - i + n$
5. $\sum_{d \in [n]} el(e,d) = 1 \quad \forall e \in [m]$
6. $\sum_{r \in [2n]} er(e,r) = 1 \quad \forall e \in [m]$

Constraint (1) expresses that every node has to be assigned to a layer. (2) ensures a feasible layering. (3) binds the edge length variables el. (4) binds the edge reversal variables er. (5) and (6) are additional constraints requiring exactly one index to be set for each row of the el and er variables.

Comments. The relaxed model allows for a trivial solution of $l(v,i) = \frac{1}{2} \; \forall v \in V$ and any two $i \in V$ with objective value of 0. This makes the model in its current form unpractical.

The layer count can be restricted by adapting constraint, replacing the n by the desired bound (1). The number of nodes per layer (without consideration of dummy nodes) can be restricted to at most $\mathcal{B}$ nodes by adding a constraint of the form $\sum_{v \in V} l(v,x) \leq \mathcal{B} \; \forall x \in [n]$.

4 Evaluations

This section presents a series of evaluations that are structured as follows.

First, we support the idea to combine the first two phases of the layer-based approach by showing that the quality of the results from separately applying the phases is not predictable and can vary significantly. Second, we determine the quality of GLay's results compared to existing methods using randomly generated graphs. We extend these evaluations to ϵ-GLay to eliminate the influence of varying numbers of reversed edges. Fourth, a subset of the AT&T graphs is examined for which current methods yield unfavorable aspect ratios.

Before we present the actual results, we describe the used methods for the first two phases of the layer-based approach, the test setups, and the used test graphs.

Cycle Breaking Phase. In case the input graph is cyclic, we use a popular and often used heuristic by Eades et al. [7]. It runs in linear time, and performs well on sparse graphs.

Exclusively for evaluation purposes, we apply a second method that uses an IP formulation to find the MFAS, i. e. the optimal solution. As we neither use this method for execution time measurements nor in practice, its runtime is irrelevant.

Layering Phase. This phase is the main focus of this paper. We use the network simplex method presented by Gansner et al. alongside the newly presented approaches in the previous section (GLay and ϵ-GLay). The network simplex method minimizes the total edge length, i. e. the number of dummy nodes, and runs fast in practice even though its runtime is not proven to be polynomial [9]. Healy and Nikolov [13] empirically found the minimization of the number of dummy nodes to be the best goal for the layering phase when a compact drawing is desired and no bounds on the height and width of the drawing are given. This makes it a good candidate for comparison as the methods we present here minimize the same objective.

In order to avoid ambiguity, we name the different combinations of evaluated cycle breaking and layering phases as follows.

EaGa: Cycle breaking heuristic of Eades et al. and layering of Gansner et al.

OptGa: Optimal cycle breaking and layering of Gansner et al.

GLay: Combined cycle breaking and layering as introduced in Section 3.1. We say ω_{len}-ω_{rev}-GLay to further specify the used weights.

ϵ-GLay: The number of reversed edges is computed by the cycle breaking heuristic of Eades et al., the decision which edges to reverse, however, is made by GLay.

To solve the optimization problems we used CPLEX 12.6 and executed the evaluations on a server with an Intel Xeon E5540 CPU and 24 GB memory.

Test Graphs. Our new approach is originally intended to improve the drawings of graphs with a large height and relatively small width, hence unfavorable aspect ratio.

Nevertheless, we first evaluate the approach using a set of 180 randomly generated graphs with 17 to 60 nodes and an average of 1.5 edges per node to test its feasibility in the general case.

Second, we filtered the well-known AT&T graphs[1] based on aspect ratio and selected 148 graphs that have 20 or more nodes and a drawing[2] 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 GLay 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 GLay 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.

Metrics. The layer-based approach is defined as a pipeline of several independent steps. On the one hand this imposes a clear hierarchy on the importance of aesthetic criteria (e. g. a uniform edge direction is more important than edge crossings), on the other hand early phases have to cope with partial information when it comes to predicting the quality of the final drawing. The actual dimensions of the drawing are not available until termination of the final edge routing phase

After the layering phase four metrics are available: The number of reversed edges, the number of dummy nodes, the number of layers, and the maximal number of nodes in a layer. The latter three metrics allow an estimation of the area and aspect ratio of the final drawing.

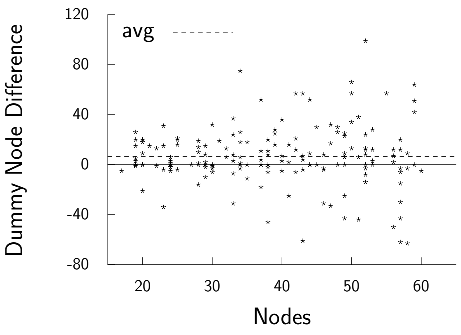
During our evaluations we first solely base on these incomplete information and later compare with the quality of the final drawing. This is for two reasons. First, we are interested in the gap between estimation and final result. Second, when searching for new methods for early phases it is necessary to know which of the metrics that are available at that point in time allow the best prediction and are thus trustworthy.

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

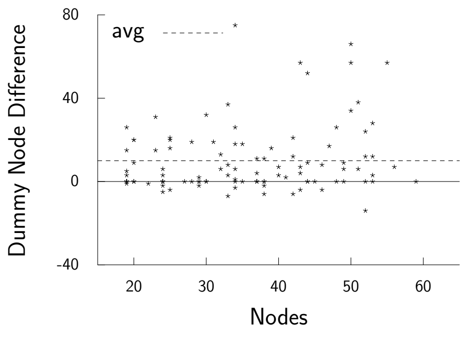
[2]Created using BK and POLY as introduced in Section 4.2.1

	Min.	Avg.	Max.
Node count	17	38.12	60
Edge count	30	60.22	90
Heuristic rev.	0	3.07	8
Optimal rev.	0	2.43	7

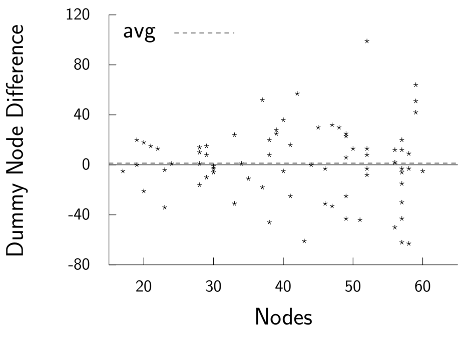
(a) Relative number of reversed edges



(b) Difference in dummy nodes between the heuristic and optimal method



(c) Detailed plot for graphs where the heuristic and optimal methods found a FAS of the same size



(d) Detailed plot for graphs with smaller optimal FAS

Figure 4.1. Evaluation results for random graphs comparing the number of reverse edges and resulting number of dummy nodes after the layering phase. For cycle breaking a heuristic method by Eades et al. and an optimal IP were used. Layering was performed using the approach by Gansner et al. Each dot represents the result for one graph instance. Positive values indicate fewer dummy nodes when using the optimal cycle breaking method.

4.1 Cycle Breaking

The underlying idea of the layer-based approach is to highlight a direction or the overall flow of a graph. Therefore it is often desirable to have as few edges pointing backward as possible, which corresponds to finding a minimum FAS. As this is an NP-complete problem, often heuristics are applied.

However, in some settings the desire for compactness outweighs the desire for a coherent direction. In the following, we examine the impact of heuristical and optimal approaches on the compactness of drawings. We will see that small differences in the number of reversed edges and the selection of the reversed edges can have a significant impact on the number of resulting dummy nodes.

Figure 4.1a illustrates that for the random graphs the heuristic reversed an average of

3.07 or about 5 % of the edges. The minimum number of edges that have to be reversed is 2.43 or about 4 %. The figures conform with the general observation that the heuristic performs well in practice.

We plotted the difference in the number of dummy nodes of EaGa and OptGa in Figure 4.1b. Each dot represents the number of EaGa dummy nodes minus the OptGa dummy nodes. Positive values indicate fewer dummy nodes after optimal cycle breaking. While on average the number of dummy nodes is slightly smaller for optimal cycle breaking, there are graphs for which the difference amounts -63 and 99. Basically, it cannot be decided for a given graph whether applying the heuristic or the optimal method yields better results.

To examine this more closely, we split the graph instances into two sets—those for which the heuristic found the optimal FAS and those with a non-optimum FAS. The results can be seen in Figure 4.1c and Figure 4.1d.

When reversing the same number of edges, the optimal strategy reversed, on average, edges that allowed for a layering with fewer dummy nodes (Figure 4.1c). This might be due to the nature of the heuristic, selecting rather disadvantageous edges for the set of test graphs. However, this is merely a speculation and out of the scope of this paper. Figure 4.1d demonstrates that finding a smaller FAS increases the number of dummy nodes significantly. This seems natural as reversing fewer edges can yield more long spanning edges in the rest of the graph.

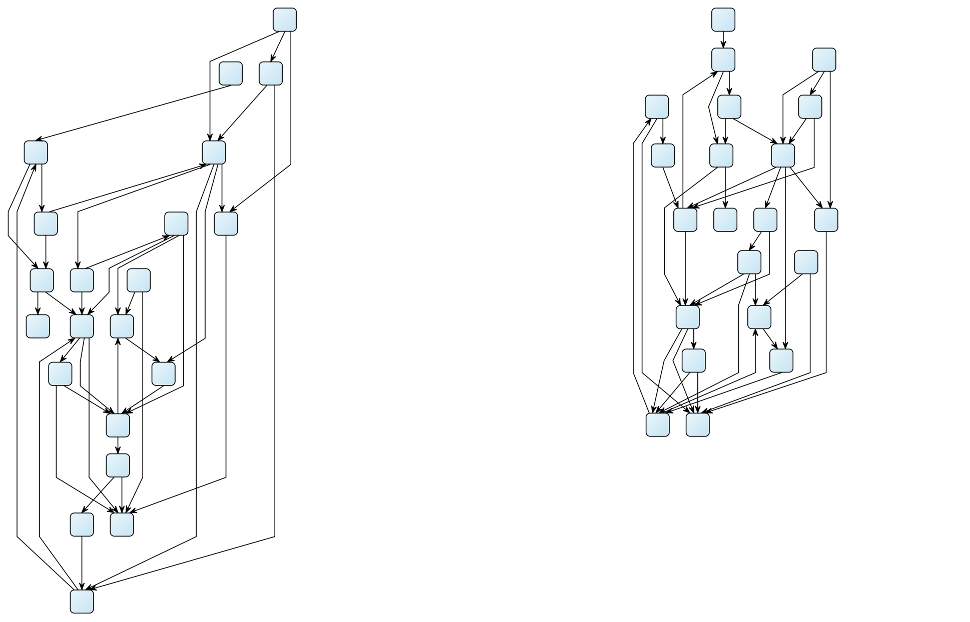
From the presented results we can conclude that for the cycle breaking phase finding the MFAS is not necessarily the best approach and that two distinct FAS of the same size might result in significantly different numbers of dummy nodes after the layering phase.

The first of the last-mentioned points was already noted by Gansner et al. [9]. They also remark that in practice graphs often have a natural direction inferred by the underlying application. In such a case a disadvantageously reversed edge might confuse the user.

4.2 Generalized Layering with Weights

We now discuss the conducted experiments for the new GLay approach, which we presented in Section 3.1. An exemplary result of the GLay approach compared to EaGa can be seen in Figure 4.2. For that specific drawing, GLay produces fewer reversed edges, fewer dummy nodes, and less area (both in width and height).

Figure 4.3 shows averaged figures for the number of reversed edges and dummy nodes plotted against the node count of the graphs. The new approach was applied to the set of random graphs with several different weights for ω_{rev} ($\omega_{\text{len}} = 1$). Figure 4.3a indicates that for $\omega_{\text{rev}} = 30$ or higher GLay produces fewer reversed edges than the classic approach. Figure 4.3b furthermore shows that for all of the used ω_{rev} values GLay yields fewer dummy nodes. This is expected, since in contrast to EaGa GLay is free in the decision which edges to reverse, while both approaches find an optimal solution in terms of dummy nodes.



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

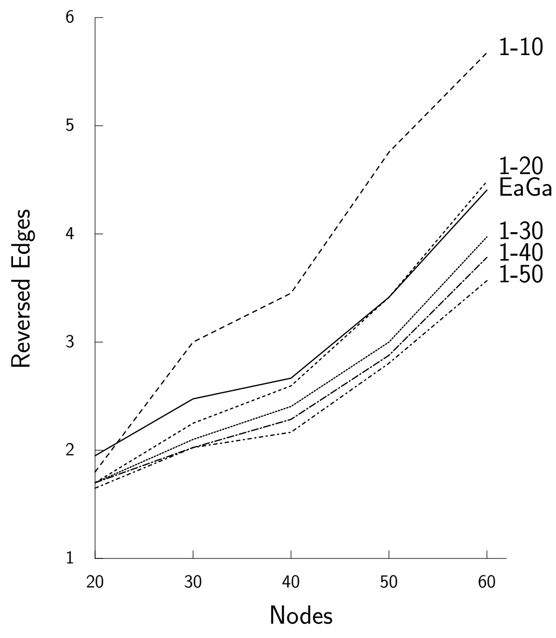
Figure 4.2. The same graph drawn with (a) EaGa (known methods as described in Chapter 4) and (b) GLay (this work). This example illustrates that GLay can perform better in both metrics: reversed edges and dummy nodes.

For certain weights GLay produces fewer reversed edges and fewer dummy nodes. While this might sound surprising in the first place, it straightforwardly follows from the facts that EaGa uses a heuristic for cycle breaking and that GLay can make more informed decisions which edges to reverse. Furthermore, it encourages combining the first two phases of the layer-based approach, which we regard a more promising approach than using optimal methods for two consecutively applied phases as evaluated in Section 4.1.

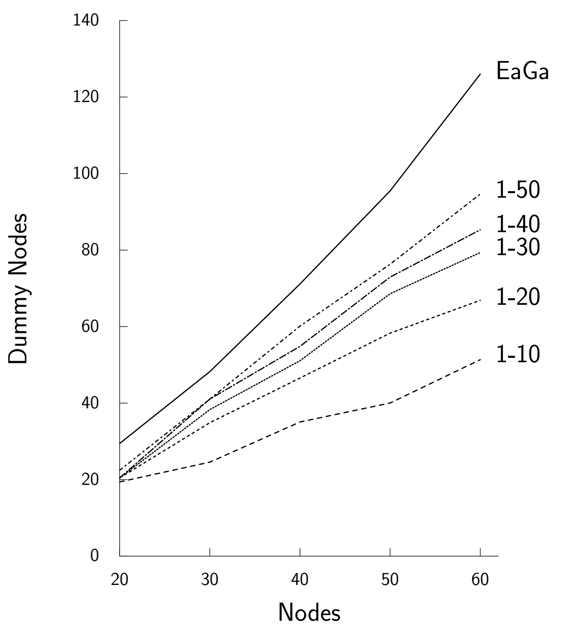
We chose to examine 1-30-GLay more closely. It results in constantly fewer reversed edges and presents a good trade-off in terms of dummy nodes. Detailed results are shown in Figure 4.4. Both subfigures show numbers where for each test graph the result of GLay is subtracted from the result of EaGa. For all but 16 of the 180 random graphs GLay produced fewer reversed edges (2.7 instead of 3.1 on average). The average in the number of dummy nodes over all graphs decreased from 78.2 to 54.6 (23.5 % on average). For one graph with 60 nodes the number of dummy nodes was reduced from 231 to 100.

Clearly, the difference in the number of dummy nodes increases with the number of nodes in the graph. Normalizing the difference by the number of nodes yields similar values for all graph sizes which suggests that GLay's performance remains stable with increasing node count.

At this stage, and with the used metrics, we can only estimate the resulting area and aspect ratio of the final drawing. Table 4.1 shows values for the estimated area of GLay, which reduce on average by 13.2 %. This is slightly less than the percentage decrease in the number of dummy nodes. Obviously, the estimated area strongly depends on the



(a)



(b)

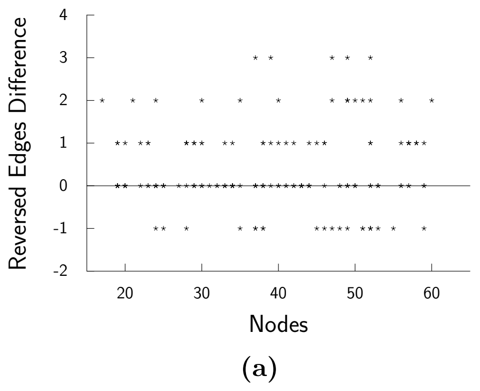
Figure 4.3. Averaged results after applying GLay with different weights for ω_{rev} to the random graphs.

maximum number of nodes in a single layer. The values for the estimated aspect ratio are corrected by a factor of 0.5, which we found to come closer to the effective aspect ratio than the plain quotient. This is since dummy nodes contribute less to the width of a layer than usual nodes; Healy and Nikolov found that for a set of directed graphs and the layering method by Gansner et al. every layer contains about 0.8 dummies nodes for each usual node [13]. Furthermore, one layer contributes often more height to the drawing than a single node contributes width; e. g. due to required space for edge routing, which in terms of weight is already encoded with dummy nodes. Independently of the correction, the estimated aspect ratios increase on average by 13.5 %. While we believe the changes described above to be improvements in terms of compactness, the number of edge crossings increases by about 10.0 %.

To assess the quality of the final drawing, and to compare with the previous estimations, we apply the remaining phases of the layer-based approach and evaluate the final drawing in the next section.

4.2.1 Final Drawing

We now investigate the final drawing after completion of the layer-based method. For that we introduce the used methods to produce the final drawing, i. e. the implementations of phases 2–5.



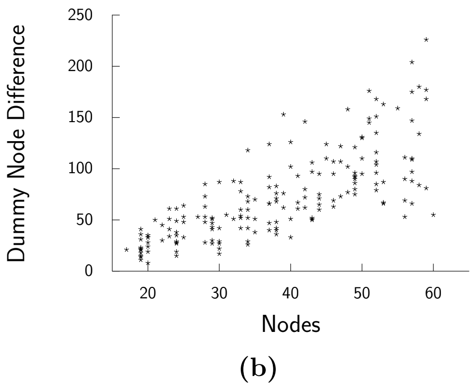


Figure 4.4. Detailed results for 1-30-GLay. Plots show the difference between the number of reversed edges and dummy nodes for EaGa and GLay, respectively. Positive values indicate lower values for GLay.

Crossing Minimization. We minimize crossings between pairs of layers using a *layer sweep* method in conjunction with the *barycenter* heuristic, as proposed by Sugiyama et al. [21].

Node Placement. We employ and compare 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 [3, 2]. It tries to straighten long edges by building chains (or *blocks*) of (dummy-) nodes. We denote this method as BK. Second, we use a method inspired by Sander [19] that similarly forms so-called *linear segments*, to draw layer-spanning edges straight. We call it LS. In general BK tends to produce fewer edge bends with a slight increase in diagram area when compared with LS [5].

Edge Routing. We compare two different strategies to route the edges; either using polylines (POLY) or orthogonal segments (ORTH). The orthogonal router is based on the methods presented by Sander [20].

Overall, this gives eight setups of the algorithm: two layering methods, two node placement algorithms, and two edge routing procedures.

The results for the final drawings are shown in Table 4.2 and are further refined by the previously introduced setups. For all setups the average area (normalized by the number of nodes) of GLay is smaller, depending on the setup, by 7–11 %. The average aspect ratio becomes closer to the desired values of 0.55 and 0.75 in three cases; in any case it increases for the selected weights. Both observations conform to the tendency of the estimated values for area and aspect ratio. Somewhat correlating to the area, the average edge length decreases as well for all setups.

We further observed that for 64 % of the graphs the tendency of the estimated area

Table 4.1. Results for random graphs and metrics that are available after the layering phase. The area is estimated by multiplying the maximum number of nodes in a layer by the number of layers. Similarly, the quotient of the latter two values is used as estimation for the aspect ratio. Crossings are averaged per edge.

		EaGa	GLay
Area (est.)	Min.	60.00	56.00
	Max.	600.00	420.00
	Avg.	222.86	187.44
Aspect ratio (est.)	Min.	0.21	0.21
	Max.	1.94	1.63
	Avg.	0.74	0.84
Crossings	Min.	0.13	0.13
	Max.	4.67	4.02
	Avg.	1.79	1.92

contradicts the tendency of the effective area,[3] and 54 % when not considering dummy nodes. In other words, the estimated area is decreased for GLay compared to EaGa and the effective area is increased for GLay (or vice versa). This clearly indicates that the estimation, as we compute it, can be misleading.

While in general the results resemble what we expected and what GLay was designed for, averaged figures hide some aspects of the overall picture. For this reason, we further inspect the distribution of the estimated area and effective area in Figure 4.5.

We use boxplots which partition the distribution of data into quartiles, showing 50 % of the data points in a box with the median value shown as a horizontal bar. Additionally, so-called whiskers extend to both sides of the box, covering 25 % of the data each. By definition the maximum extent of a whisker is 1.5 times the height of the box, any values outside the whiskers are depicted by dots and are generally assumed to be statistical outliers. In our case we cannot really speak of them as outliers since we have very heterogeneous data, but use the boxplots anyway to get a rough overview.

The right box in Figure 4.5a indicates that 75 % of the graphs are estimated to have less area when using GLay and considering dummy nodes. Without consideration of dummy nodes the results are worse than EaGa in the majority of the cases, but we believe that an estimation without dummy nodes is not representative anyway.

Figure 4.5b supports the general trend that GLay produces less area in the final drawing, however, it also yields a larger area for over 25 % of the graphs, using any setup. The two node placement algorithms do not differ significantly in their results. Nevertheless, orthogonal edge routing reduces the deviation compared to polyline routing.

From the above observations we can conclude the following. The estimated area can differ from the effective area notably. Node placement and edge routing have a stronger impact on the compactness of the drawing than generally assumed.

To understand the increase in area we tried to identify a metric that correlates to this

[3]Using BK and POLY.

Table 4.2. Results for the final drawing of a graph. Area is normalized by the node count of a graph. It can be seen that using the same edge routing and node placement strategies the average effective area and edge length for 1-30-GLay decrease compared to EaGa. The average aspect ratio increases. The most interesting comparisons are between columns where EaGa and GLay use the same strategies for the remaining steps.

		Poly				Orth			
		BK		LS		BK		LS	
		EaGa	GLay	EaGa	GLay	EaGa	GLay	EaGa	GLay
Area (norm.)	Min.	6,517	6,165	4,580	4,795	5,495	5,873	5,214	4,754
	Max.	46,403	40,467	29,111	24,523	24,451	23,245	20,075	18,848
	Avg.	20,194	18,683	12,383	11,035	13,582	12,642	10,666	9,917
Aspect ratio	Min.	0.25	0.25	0.22	0.22	0.32	0.32	0.25	0.25
	Max.	1.10	1.20	1.18	1.15	1.61	1.59	1.32	1.37
	Avg.	0.59	0.67	0.55	0.64	0.84	0.96	0.63	0.70
Avg. edge length	Min.	111.4	107.2	92.1	86.8	118.1	117.0	101.5	101.5
	Max.	512.6	441.0	420.0	324.2	388.6	412.5	370.4	358.3
	Avg.	261.5	239.9	197.7	175.4	241.4	232.0	218.5	209.4

observation. For the used graphs, however, we could not find a significant correlation between increased area and a metric that is available during the layering phase, e. g. maximum number of nodes per layer, number of layers, and edge density. Nevertheless, we list some factors that influence area and aspect ratio after the layering phase.

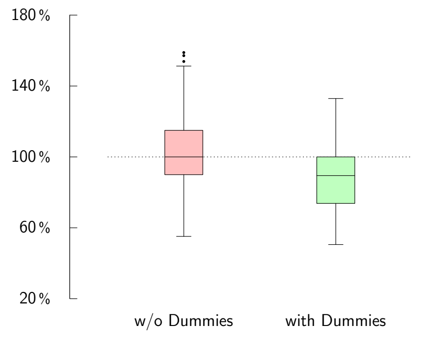
1. *Number of layers*: Affects the height of the graph.

2. *Maximum number of nodes per layer*: Can increase the overall width of the drawing, especially dummy nodes that are part of long edges that are drawn straight can push the drawing apart, reducing compactness.

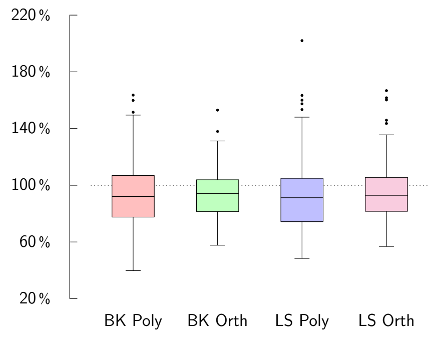
3. *Reversed edges*: Fewer reversed edges can increase the number of layers. For 1-30-GLay this was the case several times.

4. *Maximum edge density*: A lot of edges between two layers can cause the layers to be pushed apart to route the edges, increasing overall height.

In conclusion, the evaluations show that GLay is able to produce drawings with less area and better aspect ratio on average. For general graphs, the estimated area and actual area differ more than we expected and the effective area reduction is smaller than we expected. Both results need further investigation. The GLay approach specifically targets drawings with bad aspect ratio due to longest paths and allows to chose weights such that the drawing conforms to certain requirements. It seems to be a promising candidate to be used in replacement of current layering strategies.



(a) Estimated Area



(b) Effective Area

Figure 4.5. Percentage change between the area a drawing produced by EaGa and GLay. A value below 100 % indicates less area for GLay. (a) shows an estimated area by multiplying the number of layers by the maximum number of nodes per layer, once with consideration of dummy nodes and once without. (b) shows the effective area after node placement and edge routing.

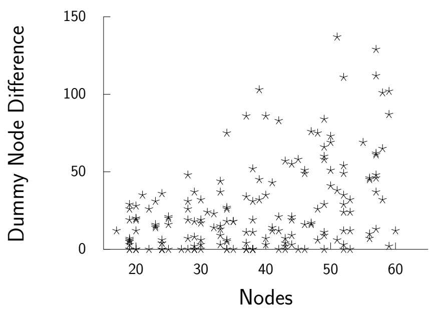


Figure 4.6. Difference in the number of dummy nodes between EaGa and ϵ-GLay. Positive values indicate a better performance of ϵ-GLay.

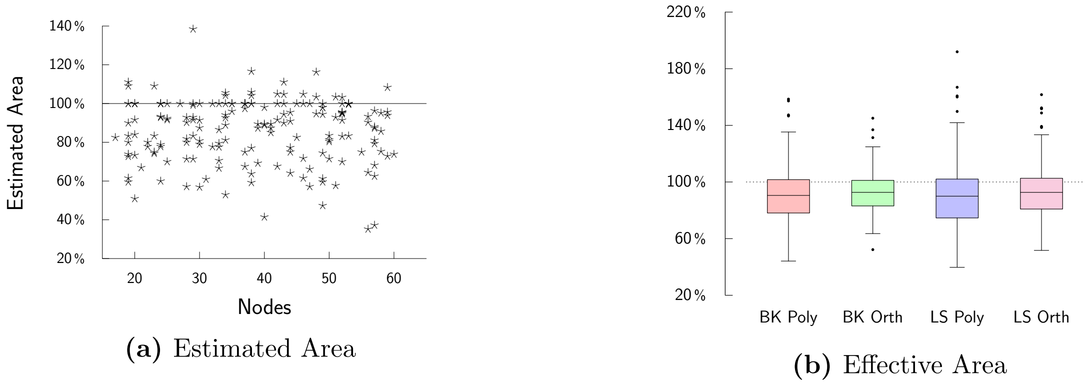
4.2.2 Weight Selection

For the evaluations of the previous sections we selected the weights ω_{len} and ω_{rev} for GLay as we deemed reasonable and selected a pair of weights for more detailed inspections that looked promising in the general case.

The ratio between the weights clearly shifts with the number of edges and nodes in a graph. Larger graphs are more likely to require more dummy nodes in relation to the number of reversed edges. Thus, it seems natural to couple the weight selection to characteristics of the graph instances.

4.3 Generalized Layering with Fixed FAS Size

In the GLP we try to optimize two objectives at the same time (number of reversed edges and number of dummy nodes), which makes the problem hard to solve, especially



(a) Estimated Area

(b) Effective Area

Figure 4.7. (a) Percentage change of the estimated area. Values below 100 % represent a decreased area for ϵ-GLay. (b) Percentage change between the area a drawing produced by EaGa and ϵ-GLay. A value below 100 % indicates less area for ϵ-GLay.

if a good trade-off between the two objectives is not known. To overcome this problem we introduce a variation of GLay named ϵ-GLay, where we fix the number of edges to reverse beforehand. This way we remove one objective function but preserve the advantage to freely select edges to reverse during the layering phase. To determine a reasonable value we apply the greedy heuristic by Eades et al. and use the size of the computed FAS to add a constraint to ϵ-GLay (see Section 3.2).

As expected, Figure 4.6 shows that for every graph of the random graphs set ϵ-GLay produces either fewer dummy nodes or the same number of dummy nodes. On average, ϵ-GLay has 25.8 fewer dummy nodes. Figure 4.7a depicts that the estimated area is equal or smaller for 162 of the 180 graphs. Compared to GLay's results on estimated area (cf. Figure 4.5) this is a superior result. It supports our claim that several of the graphs with increased area were layered with fewer reversed edges by GLay as opposed to EaGa.

Again, we executed the eight combinations of node placement and edge routing introduced in Section 4.2.1, this time with ϵ-GLay instead of GLay though. The results are similar to Table 4.2 with the difference that in all cases (area, aspect ratio, average edge length) ϵ-GLay performs on average slightly better than the previously used 1-30-GLay. This supports our assumption that the varying number of reversed edges during the GLay experiments (i. e. GLay produced less reversed edges than EaGa for several graphs) has a negative impact on the metrics of the final drawing. This is also supported by Figure 4.7b which shows boxplots analogous to Figure 4.5b. The height of the boxes reduces and the boxes shift slightly to the bottom. This indicates that a larger number of final drawings shows area improvements than before. Nevertheless, over 25 % of the graphs show a poorer area using ϵ-GLay, as opposed to EaGa, which in turn confirms our finding that the number of reversed edges is not the only candidate to influence the area of the final drawing.

4.4 AT&T Graphs

Additionally to the randomly generated graphs used in the previous sections we selected a subset of the well-known AT&T graphs, consisting of graphs with comparatively long longest paths (height) and a rather stretched aspect ratio. Since the graphs are acyclic the cycle breaking phase of the layered approach is not required and current layering algorithms cannot improve the height.

The GLay approach, however, can freely reverse edges and hereby change the height and aspect ratio. We applied EaGa and GLay with the same weights as in the previous sections to this set of graphs, the results of which can be seen in Table 4.3. Clearly, EaGa has no reversed edges as all graphs are acyclic. 1-10-GLay starts with an average of 2.74 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 141.30 for EaGa to 39.91 for 1-10-GLay. As expected, more reversed edges yield more edge crossings.

Moreover, we recorded three metrics on the final drawing after applying the node placement strategy by Brandes and Köpf and polyline edge routing. The average height and average area of the drawings decrease with an increasing number of reversed edges. For 1-10-GLay the average height and area are 38.2 % and 33.8 % smaller, respectively. The aspect ratio changes from an average of 0.20 for EaGa to 0.34 for 1-10-GLay.

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. We noticed during the analysis of the results that the set of test graphs still contains several path-like graphs with only one or two branches from the longest path. These graphs bias the results for reasons described in the paragraph on the used test graphs of Chapter 4. Furthermore, the observed 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.

As opposed to the random graphs, we did neither compare different node placement and edge routing methods nor did we apply ϵ-GLay to the AT&T graphs, mainly because the latter contain several graphs with more than 90 nodes which slows down the IP model. Nevertheless, we do not expect that the results would provide new insights in addition to our findings of the previous sections.

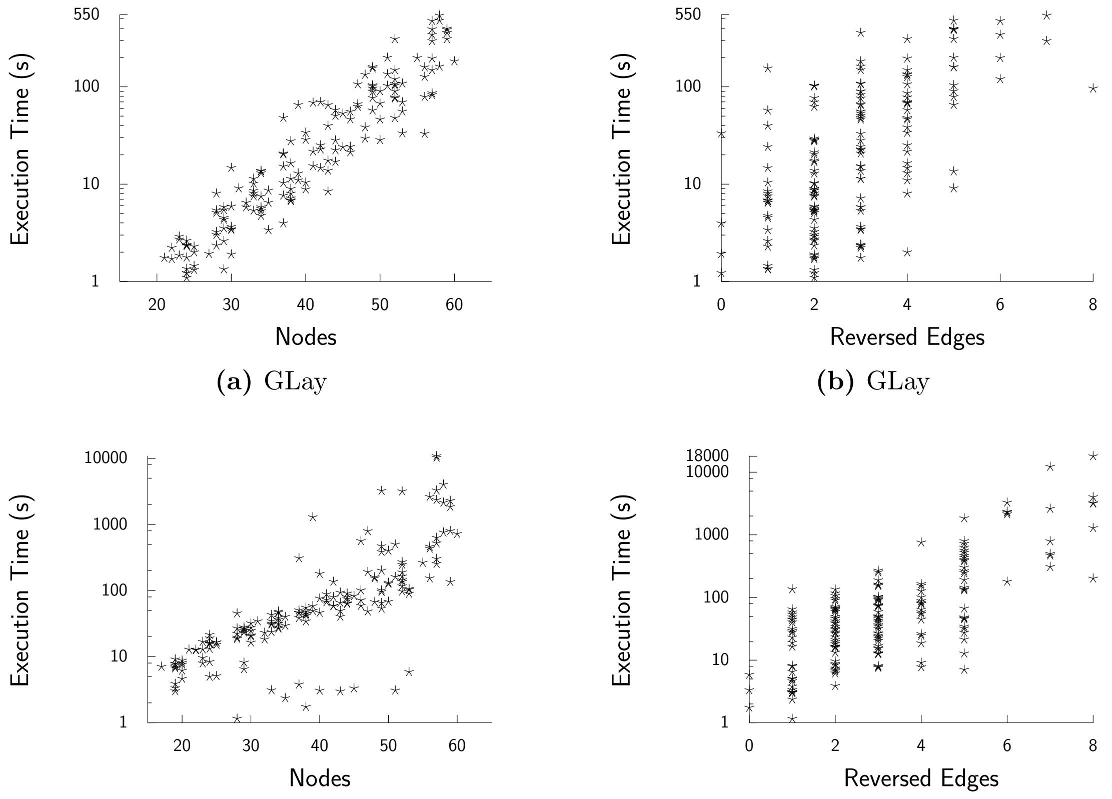
4.5 Execution Times

Figure 4.8 shows four plots of the execution times of the GLay and ϵ-GLay IPs, for each IP once plotted against the number of nodes in a graph and once for the number of reversed edges in the layered graph.

GLay. The execution times for GLay vary between 476ms for a graph with 19 nodes and 541s for a graph with 58 nodes and steadily increase with the graph's node count.

Table 4.3. Average values for different layering strategies employed to the AT&T graphs with an undesirable aspect ratio. Different weights are used for GLay as specified in the head column.

	1-10	1-20	1-30	1-40	1-50	EaGa
Reversed edges	2.74	1.47	1.02	0.72	0.56	0
Dummy nodes	39.9	55.5	65.7	75.7	82.5	141.3
Crossings	1.46	1.23	1.13	1.06	1.00	0.86
Avg height	1,068	1,224	1,334	1,409	1,469	1,727
Area	587,727	622,838	641,581	660,842	695,494	874,374
Aspect ratio	0.34	0.28	0.24	0.23	0.22	0.20



(a) GLay

(b) GLay

(c) ϵ-GLay

(d) ϵ-GLay

Figure 4.8. Execution times of the GLay and ϵ-GLay IPs when applied to the random graphs.

This is impracticable for interactive tools that rely on automatic layout, but is fast enough to collect optimal results for medium sized graphs.

In Figure 4.8a the execution time is plotted against the number of reversed edges. With an increased number of reversed edges an increased execution time can be observed, however, this corresponds to the fact that larger graphs often require more edges to be reversed.

ϵ-GLay. Initially, we suspected the runtime of ϵ-GLay to decrease compared to GLay as we add an additional constraint to the model. However, it turned out that the execution time increases significantly. Nevertheless, we believe this version of the problem to be a valuable starting point for further evaluations. Especially regarding our plans to develop a heuristic the restriction to one objective can be an advantage.

The fastest execution was about 200ms for a graph with 19 nodes and 0 reversed edges. For a single graph instance with 57 nodes and 8 reversed edges the model did not finish within the specified time limit of 3 hours (the longest time GLay required is just over 9 minutes). Interestingly, there are several graphs with up to 53 nodes that require less than 10s.

Contrary to GLay, Figure 4.8c suggests that the execution time of ϵ-GLay is coupled with the number of reversed edges. Looking at the results in detail, ϵ-GLay executes faster for all graphs with 0 reversed edges. Reversing no edges reduces the problem basically to the layering formulation introduced by Gansner et al. For certain graphs with 1–3 reversed edges ϵ-GLay is faster, and for 4 or more reversed edges it is always slower.

5 Discussion

In this paper we address problems with current methods for the first two phases of the layer-based approach. We argue that separately performing cycle breaking and layering is disadvantageous when aiming for compactness, especially for graphs with a high ratio between height and width.

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 (they can never increase). While the number of dummy nodes only allows for an estimation of the area, the effective area of the final drawing is reduced as well. Furthermore, graph instances for which current methods yield unfavorable aspect ratios can easily be improved.

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 after the layering phase can vary significantly from the effective values of the final drawing.

Our evaluations suggest that besides the layering phase the node placement phase and edge routing phase have a significant impact on the compactness of a graph's drawing. We propose to further investigate this in two ways. First, the latter two phases should be improved in terms of compactness, e. g. the maximization of straight edges during node placement could be relaxed to disregard edges that, drawn straightly, would highly increase the area. Second, correlations between the layering phase and later phases, and vice versa, are to be inspected as to what kind of layering lends itself well to a compact node placement. For instance, balanced layerings, that is layerings where most layers contain about the same number of nodes, might allow more compact node placement than unbalanced ones.

As a final example of the usefulness of the new approach consider Figure 5.1. In (b) GLay was applied to only the top-most level of this compound graph which consists of 15 nodes. The drawing is more compact and the overall layout took about two seconds (including our IP).

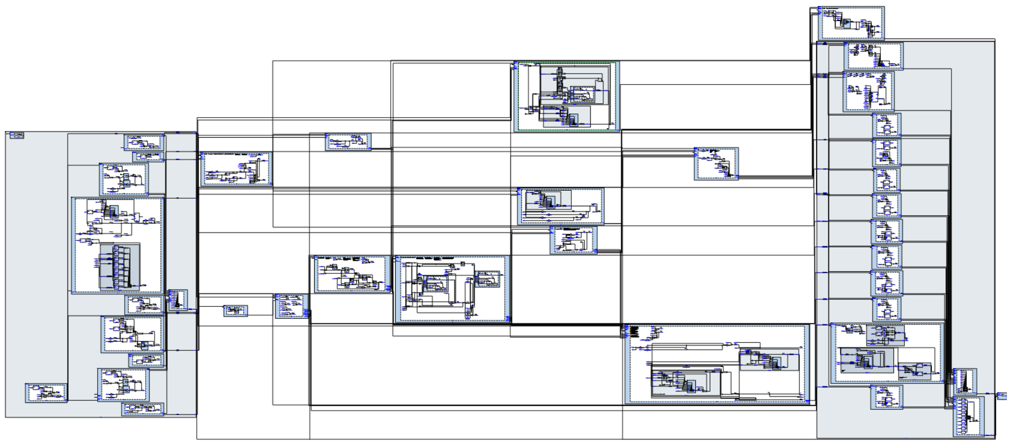
Future Research. We see several possibilities to proceed from here.

First, in order to make the new method usable in productive tools, heuristics with a reasonable execution time and quality are required. We plan to evaluate whether heuristics for the well-studied LAP can be modified to solve either the GLP or its variation ϵ-GLP. In either case it is necessary to at least allow a bound on the height of the layering to be able to improve the aspect ratio of path-like graphs. Even if such a heuristic performs worse on average than the method by Gansner et al. it could still improve on those graphs with excessive height.

Second, the presented IP models can be inspected further. For example, either to include additional features such as bounds on width and height (with consideration of dummy nodes) or to improve execution time by adding further constraints.

Third, another interesting topic is whether it is possible to target a certain aspect ratio by means of an objective function in one of the IPs. A difficulty here is to find a sound estimation of the aspect ratio of the final drawing.

Acknowledgements. We thank Chris Mears for support regarding the IP formulation. We further thank Fritz Bökler, Tim Dwyer, Carsten Gutwenger, and Petra Mutzel for valuable discussions.


(a) EaGa + LS

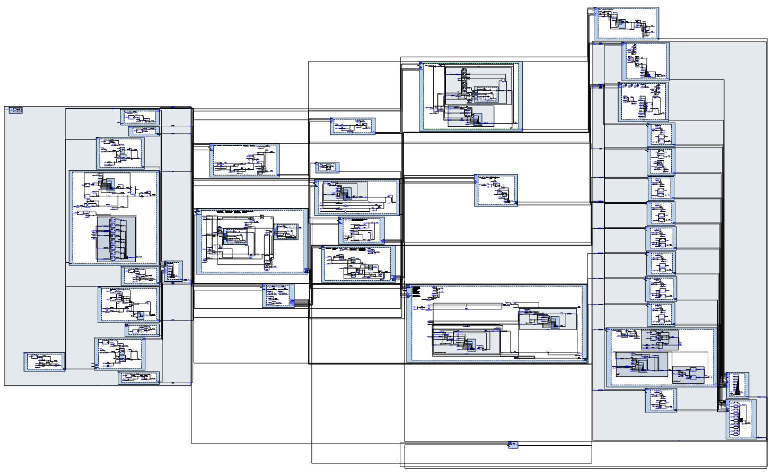

(b) 1-10-GLay + LS

Figure 5.1. Two drawings of a graph from an industrial context at the same scale level. The graph is a compound graph, i. e. nodes can contain further nodes. The layout configuration only differs for the top-most compound nodes in that the layering task is performed either by (a) Gansner et al.'s method [9] or by (b) GLay (this work).

Bibliography

[1] Radoslav Andreev, Patrick Healy, and Nikola S. Nikolov. Applying ant colony optimization metaheuristic to the DAG layering problem. In *Proceedings of the Parallel and Distributed Processing Symposium (IPDPS'07)*, pages 1–9, 2007.

[2] Ulrik Brandes and Boris Köpf. Fast and simple horizontal coordinate assignment. In *Proceedings of the 9th International Symposium on Graph Drawing (GD'01)*, volume 2265 of *LNCS*, pages 33–36. Springer, 2002.

[3] Christoph Buchheim, Michael Jünger, and Sebastian Leipert. A fast layout algorithm for k-level graphs. In *Proceedings of the 8th International Symposium on Graph Drawing (GD'00)*, volume 1984 of *LNCS*, pages 229–240. Springer, 2001.

[4] Alberto Caprara, Adam N. Letchford, and Juan-José Salazar-González. Decorous lower bounds for minimum linear arrangement. *INFORMS Journal on Computing*, 23(1):26–40, 2011.

[5] John Julian Carstens. Node and label placement in a layered layout algorithm. Master's thesis, Christian-Albrechts-Universität zu Kiel, Department of Computer Science, September 2012.

[6] P. Z. Chinn, J. Chvtalov, A. K. Dewdney, and N. E. Gibbs. The bandwidth problem for graphs and matrices – a survey. *Journal of Graph Theory*, 6(3):223–254, 1982.

[7] Peter Eades, Xuemin Lin, and W. F. Smyth. A fast and effective heuristic for the feedback arc set problem. *Information Processing Letters*, 47(6):319–323, 1993.

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

[9] 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.

[10] Michael R. Garey and David S. Johnson. *Computers and Intractibility: A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co, New York, 1979.

[11] Carsten Gutwenger, Reinhard von Hanxleden, Petra Mutzel, Ulf Rüegg, and Miro 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.

[12] Patrick Healy and Nikola S. Nikolov. A branch-and-cut approach to the directed acyclic graph layering problem. In *Proceedings of the 10th International Symposium on Graph Drawing (GD'02)*, volume 2528 of *LNCS*, pages 98–109. Springer, 2002.

[13] Patrick Healy and Nikola S. Nikolov. How to layer a directed acyclic graph. In *Proceedings of the 9th International Symposium on Graph Drawing (GD'01)*, volume 2265 of *LNCS*, pages 563–566. Springer, 2002.

[14] Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller and James W. Thatcher, editors, *Complexity of Computer Computations (Proceedings of a Symposium on the Complexity of Computer Computations, March, 1972, Yorktown Heights, NY)*, pages 85–103. Plenum Press, New York, 1972.

[15] E. Kubicka and A. J. Schwenk. An introduction to chromatic sums. In *Proceedings of the 17th Conference on ACM Annual Computer Science Conference*, CSC '89, pages 39–45, New York, NY, USA, 1989. ACM.

[16] Lev Nachmanson, George Robertson, and Bongshin Lee. Drawing graphs with GLEE. In Seok-Hee Hong, Takao Nishizeki, and Wu Quan, editors, *Graph Drawing*, volume 4875 of *LNCS*, pages 389–394. Springer Berlin Heidelberg, 2008.

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

[18] 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.

[19] Georg Sander. A fast heuristic for hierarchical Manhattan layout. In *Proceedings of the Symposium on Graph Drawing (GD'95)*, volume 1027 of *LNCS*, pages 447–458. Springer, 1996.

[20] Georg Sander. Layout of directed hypergraphs with orthogonal hyperedges. In *Proceedings of the 11th International Symposium on Graph Drawing (GD'03)*, volume 2912 of *LNCS*, pages 381–386. Springer, 2004.

[21] 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.