# How to Layer a Directed Acyclic Graph

Patrick Healy and Nikola S. Nikolov

CSIS Department, University of Limerick, Limerick, Republic of Ireland,
fax +353-61-202734
{patrick.healy,nikola.nikolov}@ul.ie

**Abstract.** We consider the problem of partitioning a directed acyclic graph into layers such that all edges point unidirectionally. We perform an experimental analysis of some of the existing layering algorithms and then propose a new algorithm that is more realistic in the sense that it is possible to incorporate specific information about node and edge widths into the algorithm. The goal is to minimize the total sum of edge spans subject to dimension constraints on the drawing. We also present some preliminary results from experiments we have conducted using our layering algorithm on over 5900 example directed acyclic graphs.

## 1 Introduction

The layering problem for directed acyclic graphs (DAGs) arises as one of the steps of the classical Sugiyama algorithm for drawing directed graphs [6]. If the nodes of a DAG are not pre-assigned to specific layers then it is necessary to separate them into such layers in order to draw the DAG in Sugiyama fashion. We call an algorithm that finds a layering of a DAG a *layering algorithm*. Normally a layering algorithm must find a layering of a DAG subject to certain aesthetic criteria important to the final drawing. While these may be subjective, some are generally agreed upon [4]: the drawing should be compact; large edge spans should be avoided; and, the edges should be as straight as possible. Compactness can be achieved by specifying bounds $W$ and $H$ on the width and the height of the layering respectively. Short edge spans are desirable aesthetically because they increase the readability of the drawing but also because the forced introduction of dummy nodes, when an edge spans multiple layers, degrades further stages of drawing algorithms. The span of edge $(u, v)$ with $u \in V_i$ and $v \in V_j$ is $i - j$. Further, the dummy nodes may also lead to additional bends on edges since edge bends mainly occur at dummy nodes.

At present there are three widely-used layering algorithms which find layerings of a DAG subject to *some* of the above aesthetic criteria. They all have polynomial running time: the *Longest Path* algorithm finds a layering with minimal height [4]; the *Coffman-Graham* algorithm finds a layering of width at most $W$ and height $h \leq (2 - 2/W)h_{min}$, where $h_{min}$ is the minimum height of a layering of width $W$ [3]; and the ILP algorithm of *Gansner et al.* finds a layering with minimum number of dummy nodes [5]. An upper bound on the width of the layering can be specified only in the Coffman-Graham algorithm. In the classical version of the Coffman-Graham algorithm the width of a layer is considered to be the number of real nodes the layer contains while neglecting the introduced dummy nodes. The algorithm can easily be modified to take into account the widths of the real nodes, but the width of the final drawing may still be much greater than expected, because of the contribution of the dummy nodes to it. The algorithm of Gansner et al. usually results in compact layerings, but the dimensions of the drawing are not controlled and they may be undesirable.

After introducing some basic definitions related to the layering problem we compare the performance of the existing layering algorithms on over 5900 example DAGs in Section 3. Then in Section 4 and Section 5 we introduce a new approach to the layering problem based on Integer Linear Programming (ILP) and identify a set of valid inequalities (some of which define facets) of the associated layering polytope. This approach allow us to construct a new ILP layering algorithm which we study and compare to Gansner et al.'s layering algorithm in Section 6. In Section 7 we discuss the results of this work and suggest further research directions.

## 2 Basic Definitions

**Definition 1.** *Given a DAG $G = (V, E)$, where each node $v \in V$ has a positive width $w_v$, a layering of $G$ is a partition of its node set $V$ into disjoint subsets $V_1, V_2, \ldots, V_h$, such that if $(u, v) \in E$ where $u \in V_i$ and $v \in V_j$ then $i > j$. A DAG with a layering is called a layered digraph.*

**Definition 2.** *The height of a layered digraph is the number of layers, $h$.*

**Definition 3.** *The width of layer $V_k$ is traditionally defined as $w(V_k) = \sum_{v \in V_k} w_v$ and the width of a layered digraph (layering) is $w = \max_{1 \leq k \leq h} w(V_k)$.*

Layered digraphs are conventionally drawn so that all nodes in layer $V_k$ lie on the horizontal line $y = k$ and all edges point downwards. In the process of drawing a layered digraph when an edge spans multiple layers, it is common to introduce dummy nodes with in- and out-degree 1 in the intermediate layers.

We denote by $d^-(v)$ and $d^+(v)$ the in- and out-degree of node $v \in V$, respectively. For $G = (V, E)$ with unitary edge lengths, define $l_p(v)$ to be the length of the longest path from any node $u$ to $v$ where $d^-(u) = 0$. Similarly, define $l_s(v)$ to be the longest path from $v$ to any node $u$ where $d^+(u) = 0$. That is, the values $l_p(v)$ and $l_s(v)$ refer, respectively, to the length of the longest path from any predecessor to $v$ and to the length of the longest path from $v$ to any successor. Let the node set $V$ of $G$ be constrained to be partitioned into at most $H$ layers. Then for each node $v \in V$ there is a set of consecutive layers where the node can be potentially placed in if all the edges are required to point downwards. The following three definitions describe this set.

**Definition 4.** *The roof of node $v$ is the number of the highest layer node $v$ can be placed in. We denote the roof of $v$ by $\rho(v)$, i.e. $\rho(v) = H - l_p(v)$.*

**Definition 5.** *The floor of $v$ is the lowest level node $v$ can be placed in. We denote the floor of $v$ by $\varphi(v)$, i.e. $\varphi(v) = l_s(v) + 1$.*

**Definition 6.** *The layer span of node $v$ is $L(v) = \{k \in \mathbb{N} : \varphi(v) \leq k \leq \rho(v)\}$. That is, $L(v)$ refers to the set of layers in which node $v$ can be potentially placed if all the edges are required to point downwards.*

By definition the roof and the layer span of node $v$ depend on the upper bound on the number of layers $H$. We do not include $H$ in the notations of the roof and the layer span for simplicity of notation. Normally, it is clear from the context what is the value of $H$.

## 3 Performance of Existing Layering Algorithms

In this section we look at the behavior of the algorithms described earlier on a variety of inputs. As motivation for what is to follow we look at the output of the three algorithms on a specific graph, firstly. Then we consider their aggregate performance over a range of measures.

### 3.1 Three Layerings of a Graph

Figure 1 illustrates how the same DAG, Grafo1012.22 from the graph database introduced by Di Battista et al. [1], is layered by the three algorithms described earlier.

The layering in Figure 1(a) is the output of the Longest Path algorithm and it clearly shows its weakness: the width of the bottom layers is much larger than the width of the top layers. The layering in Figure 1(b) is the output of the Coffman-Graham algorithm with an input parameter $W = 5$ (i.e. maximum 5 nodes in a layer). As can be seen, the final width of the drawing can exceed the input parameter. The third layering in Figure 1(c) is the output of the Gansner et al.'s ILP layering algorithm. In this case the drawing has the minimum number of dummy nodes, but this algorithm does not put any bounds on the dimensions of the layering, which may result in a final drawing which does not fit the drawing area. For instance, if we need the same DAG Grafo1012.22 drawn on less than 9 layers (9 is the number of layers in Figure 1(c)) we may prefer to have the DAG layered as it is in Figure 1(d) which shows a more compact drawing of Grafo1012.22 having three dummy nodes more than the minimum.

### 3.2 Aggregate Performance of Layering Algorithms

Each of the three algorithms described above has some positive attribute: the Longest Path algorithm finds a layering of minimum height; Gansner et al.'s algorithm minimizes the number of dummy nodes; and, the Coffman-Graham algorithm permits one to specify a bound on the width of the drawing. We investigate the three algorithms' performance now on a large sample of graphs according to some accepted aesthetics. These aesthetics are 1) the area of the layering and 2) the number of dummy nodes that the algorithms introduce. We ran all three algorithms on 5911 connected DAGs from the graph database of Di Battista et al. [1]. The DAGs have node counts ranging from 10 to 100 and the average number of nodes is 48.34. The node labels in all DAGs are numbers, which makes all the nodes equally wide and allow us to set the node width equal to 1. We separated the graphs into "buckets" according to their node count, putting a graph of $n$ nodes into bucket $\lfloor n/2 \rfloor$. (A similar separation according to edge count was rejected because of the number of buckets with just a single graph.)

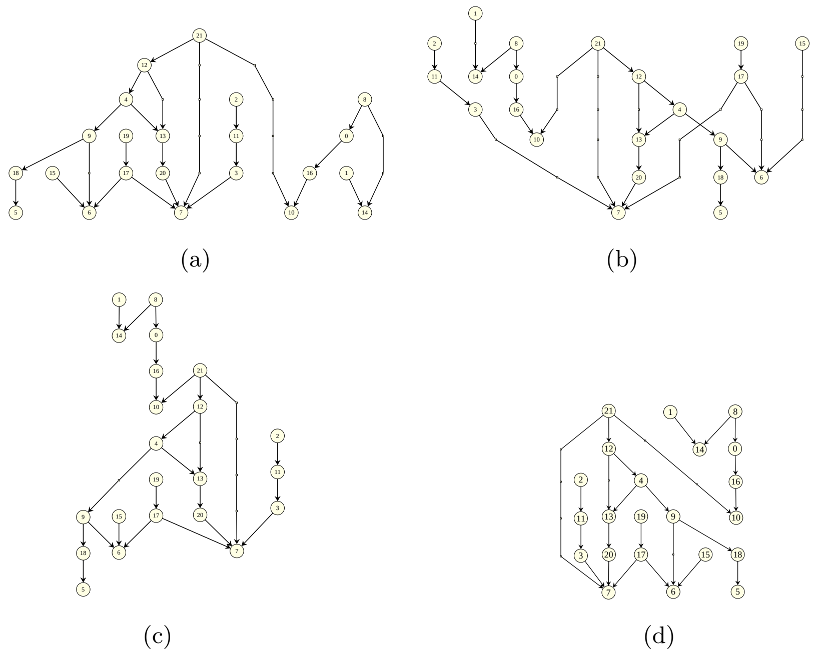
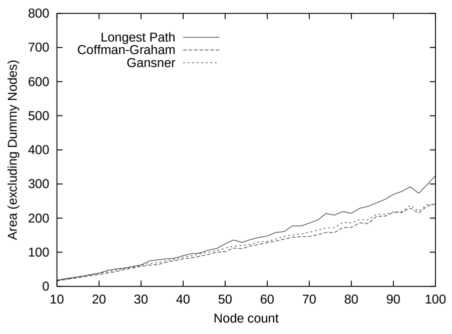


**Fig. 1.** Grafo1012.22 layered with (a) the Longest Path algorithm (b) the Coffman-Graham algorithm and (c) Gansner et al.'s algorithm. The drawing in (d) is a compact representation of Grafo1012.22 with three dummy nodes more than the minimum number of dummy nodes.

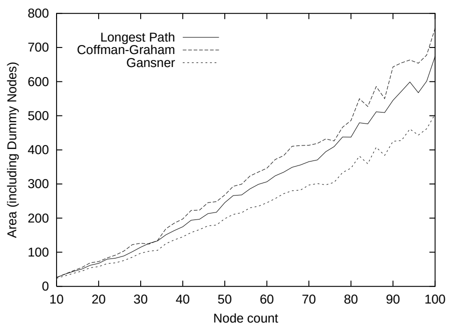
In the following, we refer to the three algorithms as Longest Path, Coffman-Graham and Gansner. For Coffman-Graham, the width bound for a graph was specified to be equal to Gansner's width for the same graph.

Figure 2(a) illustrates the three algorithms' performance according to the area aesthetic. For this figure, area was calculated to be the product of the number of layers and the width of the graph in terms of real nodes. Although Longest

Path's height is optimal, its width is so poor that it results in an increasingly poor performance. On the other hand, Gansner maintains a very close watch on Coffman-Graham in spite of there being no explicit mechanism to control its dimensions. This may be due to the following reason: the Coffman-Graham algorithm requires a bounding width, $W$, as input and the bound that we provided in all cases was the width resulting from the Gansner layering.



(a)



(b)

**Fig. 2.** Area Computed by the Layering Algorithms (a) excluding and (b) including the dummy nodes contribution to the width of the layering.

If we include the dummy nodes as part of the width of a layer then the picture changes. Figure 2(b) shows the areas of the layerings found by the three algorithms when a nominal charge of 1 is applied to dummy nodes. That is, a dummy node makes the same contribution to the width as a real node – a charge that is at the upper limit of what seems reasonable. For the larger graphs, the best area is now almost double the previous best. The Coffman-Graham algorithm has slipped to a very convincing last place however.

Although space restrictions prevents us from reporting in more detail, the aspect ratio (AR) of the resulting graphs[1] tells a similar story: ignoring dummy nodes, Gansner and Coffman-Graham behave similarly and better than Longest Path in that they have AR closer to unitary; when dummy nodes are factored into the calculations, Gansner is considerably better than the other two, which behave broadly similarly. Since Longest Path always has minimum height, its AR will be biased towards larger values. (An interesting observation about AR is that in all three algorithms, it peaks in the data at about $|V| = 75$ and then declines.)

To investigate further the apparent impact of dummy nodes on aesthetics such as area and aspect ratio, we computed both the average and maximum ratios of dummy nodes to real nodes in each of the graphs. We define these two parameters as follows.

[1] Aspect ratio is computed as the ratio of width to height.

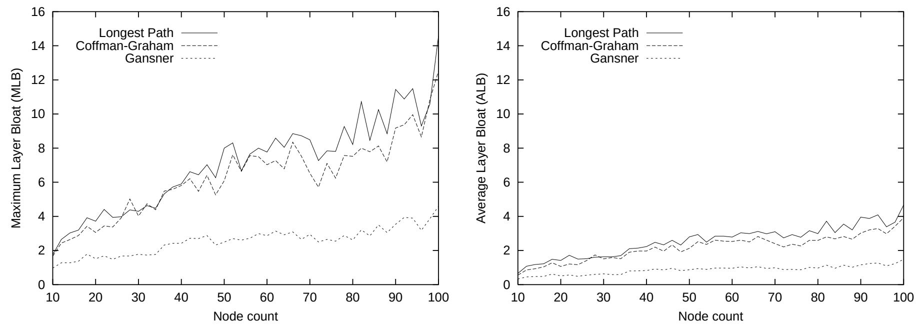
**Definition 7.** *Let $G = (V, E)$ be a layered DAG. Maximum Layer Bloat (MLB) of a layering of $G$ is the maximum ratio of the number of dummy nodes in a layer to the number of real nodes in the same layer over the layers of a graph.*

**Definition 8.** *Let $G = (V, E)$ be a layered DAG. Average Layer Bloat (ALB) of a layering of $G$ is the average of the ratios of the number of dummy nodes in a layer to the number of real nodes in the same layer over the layers of a graph.*

Table 1 summarises the maximum and average bloat values for the 5911 DAGs. A more detailed breakdown is shown in Figure 3 where it can be seen that Gansner is a clear winner: the *maximum* bloat of this algorithm behaves similar to the average of the other two algorithms. The aspect ratio and area aesthetics which are determined by the widest layer will be poorest with either Coffman-Graham or Longest Path. However, even for Gansner the number of dummy nodes is becoming significant for larger node counts.

**Table 1.** Average values of MLB and ALB for 5911 sample DAGs.

| Layering | MLB | ALB |
|---|---|---|
| Longest Path | 6.41 | 2.34 |
| Coffman-Graham | 5.76 | 2.01 |
| Gansner | 2.38 | 0.81 |



(a) (b)

**Fig. 3.** (a) Maximum and (b) Average Layer Bloats Computed by the Layering Algorithms.

The number of dummy nodes introduced by a layering algorithm impacts on later stages of hierarchical graph drawing methods so we investigate this parameter. Figure 4(a) displays the number of dummy nodes introduced by the three algorithms normalized by the number of nodes in the original graph. Gansner computes the optimal number of dummy nodes and, although growing at a super-linear rate, it easily outperforms the other two. In this context again, their performance is quite similar.

Finally, we compare the running times of the three layering algorithms in Figure 4(b). As can be seen, Longest Path has the best running time, followed by Gansner, and Coffman-Graham is the slowest, but still very fast. (We attribute the stratified nature of the plots to clock resolution.)

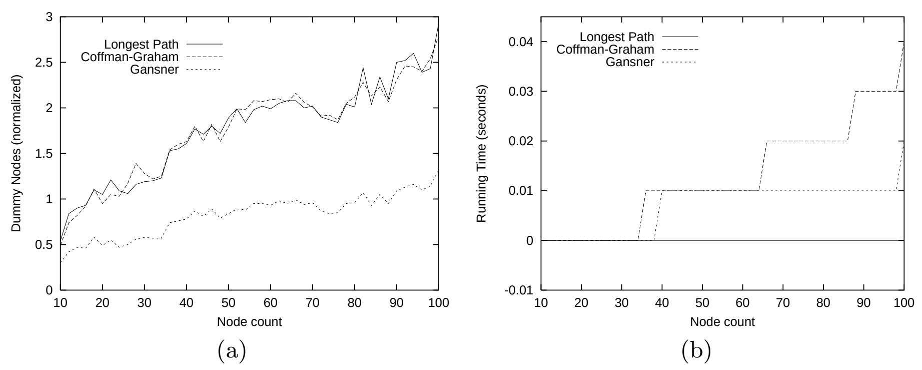


**Fig. 4.** (a) Normalized Number of Dummy Nodes introduced by the Layering Algorithms (b) Running times.

On the basis of these experiments we conclude that Gansner's algorithm is the superior of the three. However, the disadvantage in using this algorithm over the Coffman-Graham algorithm is that it is not possible to take account of variable node widths; nor is it possible to find a layering with a restriction on the width. Thus the goal of this work is to find a method for layering a DAG subject to specified maximum dimensions, where nodes and edges have variable widths. The dimensions of the layering should be true in that the width of dummy nodes should not be assumed to be negligible and the layering should minimize the sum of the edge spans (or equivalently the number of dummy nodes). We call this optimization problem *WHS-Layering*.

The width of the dummy nodes at the layering phase can be taken into account only heuristically, because it mainly depends on the last phase of Sugiyama algorithm for drawing DAGs, which is the tuning of the final position of the nodes. We see two different heuristic approaches. First, we may decide to assign a unit width to all dummy nodes and use this unit width to express the width of the other nodes of the DAG. In such case all the dummy nodes have width 1 at the layering phase. Alternatively, each edge may have different width of the dummy nodes which has to be placed on it. The experimental results we present in this paper are based on the assumption that all the dummy nodes of a DAG have unitary width.

## 4 An ILP Approach to *WHS-Layering*

Taking into account the dimensions constraints (width and height) alone makes *WHS-Layering* NP-hard since the Precedence Constrained Multiprocessor Scheduling problem can be reduced straightforwardly to it [3]. We have taken a combinatorial optimisation approach to *WHS-Layering* aiming to construct an exact ILP algorithm which finds a layering with minimum number of dummy nodes for specified upper bounds on the height and the width of the layering.

Let $G = (V, E)$ be a DAG the nodes of which have to be partitioned into at most $H > 0$ layers $V_1, V_2, \ldots, V_H$ of width at most $W > 0$. We construct a *layering DAG* or *LDAG* $\mathcal{L}_G^H = (V_\mathcal{L}, E_\mathcal{L})$ as follows. For each $v \in V$ and each $k \in L(v)$ there is a node $\lambda_{vk} \in V_\mathcal{L}$. That is, node $\lambda_{vk}$ corresponds to node $v \in V$ placed in layer $V_k$. The pair $(\lambda_{uk}, \lambda_{vl}) \in E_\mathcal{L}$ if and only if $(u, v) \in E$.

*Property 1.* Let $\mathcal{L}_G^H = (V_\mathcal{L}, E_\mathcal{L})$ be an LDAG of $G = (V, E)$, $H > 0$. If $(u, v) \in E$ then $\varphi(u) > \varphi(v)$ and $\rho(u) > \rho(v)$.

*Property 2.* Let $\mathcal{L}_G^H = (V_\mathcal{L}, E_\mathcal{L})$ be an LDAG of $G = (V, E), V = n, E = m$, $H > 0$. Then $|V_\mathcal{L}| = O(n^2)$ and $|E_\mathcal{L}| = O(mn^2)$.

**Definition 9.** *The set $F \subseteq V_\mathcal{L}$ partially represents $G$ if 1) $\lambda_{uk} \in F$, $\lambda_{vl} \in F$ and $k \neq l$ imply $u \neq v$ and 2) all the edges of $\mathcal{L}_G^H[F]$, i.e. the subgraph of $\mathcal{L}_G^H$ induced by $F$, point downwards. We call $F$ a partial layering of $G$.*

**Definition 10.** *$F \subseteq V_\mathcal{L}$ represents $G$ if $F$ partially represents $G$ and for each node $v \in V$ there is a node $\lambda_{vk} \in F$ for some $k$.*

Note that if $F$ partially represents $G$ then $\mathcal{L}_G^H[F]$ is a layered digraph, where each node $v \in V$ is represented by at most one node of $\mathcal{L}_G^H$. The pair $(V_\mathcal{L}, \mathcal{I})$, where $\mathcal{I}$ is the family of all the subsets of $V_\mathcal{L}$ which represent $G$ and induce layered digraphs of width at most $W$, is an independence system. The problem of finding a layering of $G$ on at most $H$ layers of width not greater than $W$ and minimum total sum of edge spans can be expressed as an optimization problem over the independence system $(V_\mathcal{L}, \mathcal{I})$ as follows: $\min\{C(F) : F \in \mathcal{I}\}$, where $C(F) = \sum_{\lambda \in F} c(\lambda)$ and $c(\lambda)$ is a weight associated with each node $\lambda \in V_\mathcal{L}$.

We need such a weight function $C$ with co-domain $\mathbb{R}$, so that $C(F)$ reaches its minimum at a set $F$ which induces a layered digraph with minimum total sum of edge spans. In order to do this consider the binary variables $x_{vk}$ for each node $v \in V$ and each $k \in L(v)$. Let $x_{vk}$ be 1 if node $v$ has to be placed in layer $V_k$, and 0 otherwise. Then the expression

$$\sum_{(u,v)\in E} \Big( \sum_{k=\varphi(u)}^{\rho(u)} k x_{uk} - \sum_{k=\varphi(v)}^{\rho(v)} k x_{vk} \Big) = \sum_{v \in V} \sum_{k=\varphi(v)}^{\rho(v)} k \Big( d^+(v) - d^-(v) \Big) x_{vk}$$

represents the sum of edge spans of $\mathcal{L}_G^H[F]$. If we set $c(\lambda_{vk}) = k(d^+(v) - d^-(v))$ then the minimum of the weight function would correspond to a layering with minimum total sum of edge spans. But the minimum can potentially be reached at a partial layering which does not represent $G$. To ensure that the minimum will be reached at a set that represents $G$ we set $c(\lambda_{vk}) = k(d^+(v) - d^-(v)) - M$, where $M$ is an appropriately large positive number, for instance, $M = H \times |E|$. Then the optimization problem over $(V_{\mathcal{L}}, \mathcal{I})$ takes the form

$$\min \left\{ \sum_{\lambda_{vk} \in F} [k(d^+(v) - d^-(v)) - M] : F \in \mathcal{I} \right\} .$$

We call the polytope associated with the family of subsets $\mathcal{I}$, the *graph layering polytope* and we denote it by $\mathcal{GLP}(\mathcal{L}_G^H, W)$.

In the next section we summarize the properties of $\mathcal{GLP}(\mathcal{L}_G^H, W)$.

## 5 Properties of the Layering Polytope

So far we have identified a number of nontrivial generic types of valid inequalities for the layering polytope $\mathcal{GLP}(\mathcal{L}_G^H, W)$, the most interesting of which are listed below.

*Assignment inequalities*

$$\sum_{k=\varphi(v)}^{\rho(v)} x_{vk} \leq 1 \qquad (1)$$

for all $v \in V$. These inequalities express the fact that if $F$ is a partial layering of the DAG $G$ then each node of $\mathcal{L}_G^H[F]$ corresponds to at most one node of $G$.

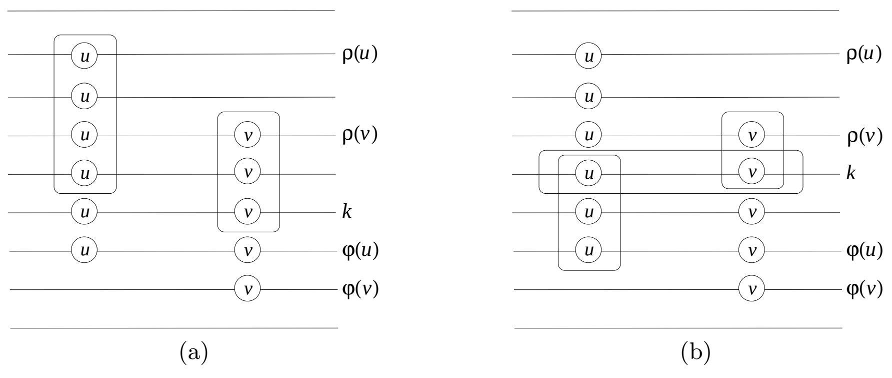


**Fig. 5.** Illustration of (a) weak RO and (b) strong RO inequalities.

*Weak relative-ordering (RO) inequalities*

$$\sum_{i=k}^{\rho(v)} x_{vi} - \sum_{i=k+1}^{\rho(u)} x_{ui} \leq 0 \tag{2}$$

with $\max(\varphi(v), \varphi(u) - 1) \leq k \leq \rho(v)$. These inequalities are valid when we have established that node $u$ must be placed above node $v$. This might be true either because there is an edge $(u, v)$ or because not placing $u$ above $v$ leads to a layering of width greater than $W$. In this case if $\sum_{i=k}^{\rho(v)} x_{vi} = 0$, then $-\sum_{i=k+1}^{\rho(u)} x_{ui} \leq 0$; and if $\sum_{i=k}^{\rho(v)} x_{vi} = 1$ then node $v$ is placed in one of the layers from $V_k$ to $V_{\rho(v)}$ and in that case, node $u$ is placed above node $v$, i.e. in one of the layers from $V_{k+1}$ to $V_{\rho(u)}$ and therefore $\sum_{i=k+1}^{\rho(u)} x_{ui} = 1$. (See Figure 5(a).)

*Strong relative-ordering (RO) inequalities*

$$\sum_{i=\varphi(u)}^{k} x_{ui} + \sum_{i=k}^{\rho(v)} x_{vi} \leq 1 \tag{3}$$

with $\varphi(u) \leq k \leq \rho(v)$. These inequalities are also valid in case we have established that node $u$ must be placed above node $v$. (See their illustration in Figure 5(b).)

*Path-augmented layer (PAL) inequalities*

$$\sum_{i=1}^{r} x_{v_i k} + \sum_{i=1}^{m_p} x_{u_i^p k} + \sum_{i=1}^{m_s} x_{u_i^s k} \leq r - 1 \tag{4}$$

where

– nodes $v_1, v_2, \ldots, v_r$ are pairwise independent (without a direct path between any two of them) which cannot be placed together into the same layer without causing the width of the layering to exceed the upper bound $W$;
– $m_p \geq 0$ and nodes $u_1^p, \ldots, u_{m_p}^p$ form a directed path $(u_{m_p}^p, \ldots, u_1^p)$ where $u_1^p$ is a common immediate predecessor to each of the nodes $v_1, \ldots, v_r$;
– $m_s \geq 0$ and nodes $u_1^s, \ldots, u_{m_s}^s$ form a directed path $(u_1^s, \ldots, u_{m_s}^s)$ where $u_1^s$ is a common immediate successor to each of the nodes $v_1, \ldots, v_r$;
– $k \in \left(\bigcap_{i=1}^{r} L(v_i)\right) \bigcap \left(\bigcap_{i=1}^{m_p} L(u_i^p)\right) \bigcap \left(\bigcap_{i=1}^{m_s} L(u_i^s)\right)$.

*Capacity inequalities*

$$\sum_{v \in V(k)} w_v x_{vk} + \sum_{(u,v) \in E} w_e \Big( \sum_{l \in L(u),\, l > k} x_{ul} - \sum_{l \in L(v),\, l \geq k} x_{vl} \Big) \leq W \tag{5}$$

where $1 \leq k \leq H$ and $w_e$ is the width of the dummy nodes placed in edge $e$. In this way we are able to localize dummy node widths for each edge of a DAG. Inequalities (5), called *capacity constraints*, restrict the width of each layer (including the dummy nodes) to be less than or equal to $W$: the first sum on the left hand side represents the contribution of the real nodes to layer $V_k$ while the second sum is the contribution of the dummy nodes. The value of the expression inside the brackets in (5) is 1 if and only if the edge $(u, v)$ spans layer $V_k$. Otherwise it is 0.

The two lemmas[2] below describe some trivial properties of the layering polytope $\mathcal{GLP}(\mathcal{L}_G^H, W)$.

**Lemma 1.** *The dimension of the layering polytope of* $G = (V, E)$ *is equal to* $\sum_{u \in V}(\rho(u) - \varphi(u) + 1)$*, so it is full dimensional. For each node* $\lambda_{vk}$ *of of* $\mathcal{L}_G^H$ *the inequalities* $x_{vk} \geq 0$ *define facets of* $\mathcal{GLP}(\mathcal{L}_G^H, W)$*.*

**Lemma 2.** *The weak RO inequalities (2) are not facet defining for the layering polytope.*

The following two theorems describe facet-defining properties of the assignment inequalities and the strong RO inequalities.

**Theorem 1.** *The assignment inequalities (1) are facet defining for the layering polytope.*

**Theorem 2.** *Let* $G = (V, E)$ *be a DAG. The strong RO (relative-ordering) inequalities (3) are facet-defining for* $\mathcal{GLP}(\mathcal{L}_G^H, W)$ *in the following two cases.*

– $(u, v) \in E$ *is a non-transitive edge.*
– $u$ *and* $v$ *are two independent nodes (without a directed path between them) and* $u$ *has to placed above* $v$ *in order for the layering to have width at most* $W$*.*

Note that the strong RO inequalities for all the non-transitive edges are sufficient to ensure that all the edges of the layered digraph point downwards.

## 6 Experimental Results

We have constructed an ILP formulation, called `ULair`, which models the layering problem *WHS-Layering* employing the assignment inequalities (1), the strong RO inequalities (3) and the capacity inequalities (5). `ULair` is solved currently by running an ILP solver directly from CPLEX 7.0 on an Intel Pentium III/Red Hat Linux 6.2 platform. As we have shown in Section 3 Gansner outperforms the other present layering algorithms aesthetically. Thus we compared `ULair`'s performance to Gansner's. We ran Gansner and `ULair` with the same 5911 DAGs,

[2] Proofs of all lemmas and theorems in this section are available upon request.

described in Section 3.2, measuring a variety of parameters of the layering solutions: MLB, ALB, maximum edge density between two adjacent layers, aspect ratio of the width and the height of the layerings as well as their area.

`ULair` has two input parameters: an upper bound on the height of the layering, $H$, and an upper bound on the width of the layering, $W$. We conducted two groups of experiments over the 5911 example DAGs.

**First group of experiments**. Firstly we ran Gansner on all the 5911 DAGs. Suppose Gansner gives a solution with height $H_G$ and width $W_G$ and $W_G \geq H_G$ (alternatively $H_G > W_G$) for a DAG $G$. Then we took $W = W_G$ and $H = W/AR$ (alternatively $H = H_G$ and $W = H/AR$), where $AR$ is the desired aspect ratio of the width and the height, as input parameters to `ULair`. We have tried two values of $AR$: the golden mean 1.618 and 1.0. In the case $AR = 1.618$ `ULair` reported solutions for 84.88% (5017 DAGs) of all the 5911 DAGs and in the case $AR = 1.0$ `ULair` reported solutions for all the DAGs except one. The values of all the parameters of the layerings we watched are slightly better than the parameters of the solutions given by Gansner, which shows that `ULair` gives the same quality of solutions when the dimensions are approximately the same as the dimensions of Gansner's solutions. In Figure 6 we present the maximum edge density between adjacent layers in Gansner's and `ULair` solutions. The edge density is normalized (i.e. divided by the total number of edges). In this group of experiments, the maximum edge density is the parameter on which Gansner and `ULair` differ most. The better values of edge density in `ULair`'s layerings are encouraging, because they suggest a more even distribution of the graph over the drawing area and perhaps to a fewer number of edge crossings at a later stage of the Sugiyama algorithm. This is a subject of further research.

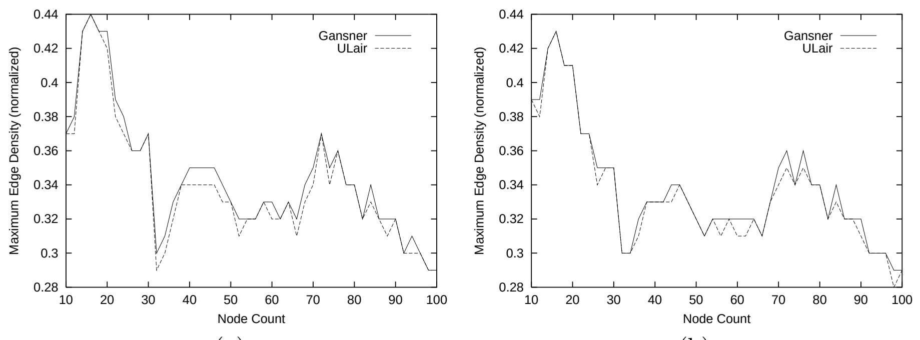


**Fig. 6.** First group of experiments: Maximum edge density when (a) $AR = 1.618$ and (b) $AR = 1.0$.

**Second group of experiments**. We then ran Gansner and `ULair` independently for the 5911 example DAGs. The input parameters of `ULair` were set according to a certain aspect ratio $AR$ of the width and the height of a layering, assuming that the nodes are distributed evenly over the layers and the expected number of dummy nodes is equal to the number of real nodes. That is, $H$ is the larger of $\sqrt{2n/AR}$ and the longest path in the DAG, and $W = H \times AR$. We performed these experiments for $AR = 1.618$ and `ULair` reported solutions for 70.39% (4161 DAGs) of all the 5911 DAGs. Here `ULair` performed better than in the previous group. Figure 7(a) compares the area and Figure 7(b) compares the edge density of the layerings of Gansner and `ULair`.

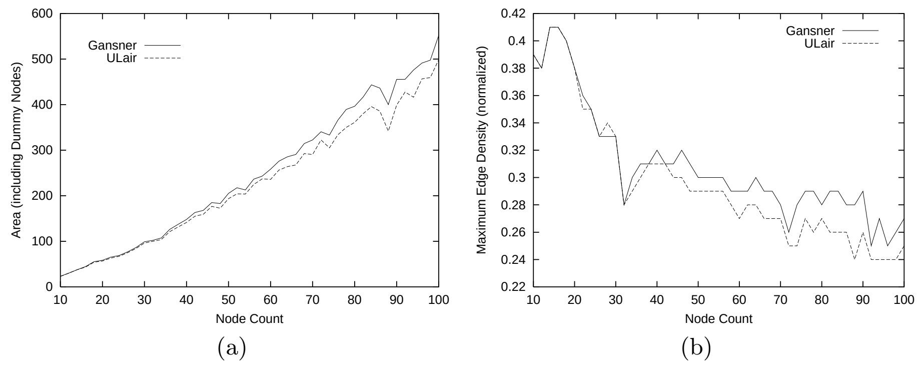


**Fig. 7.** Second group of experiments: (a) area and (b) maximum edge density of the layering solutions ($AR = 1.618$).

The running times for the two groups of experiments are presented in Figure 8. When we changed AR from 1.618 to 1.0 in the first group of experiments, the running time visibly increased (see Figure 8(a)). Smaller AR means either larger height upper bound $H$ or smaller width upper bound $W$ (because we chose $AR = W/H$). Our experience working with `ULair` has shown that when the width upper bound is constant then decreasing the height upper bound speeds up `ULair`, and when the height upper bound is constant then decreasing the width upper bound slows down the layering procedure. Since the problem under investigation is NP-hard we do not expect better running time than the existing layering algorithms which have polynomial time complexity. However, we believe that `ULair` – as it presently stands – is a better alternative for DAGs having up to 100 nodes (and possibly more), as well as in any case when the time for drawing is less important than the quality of the final picture.

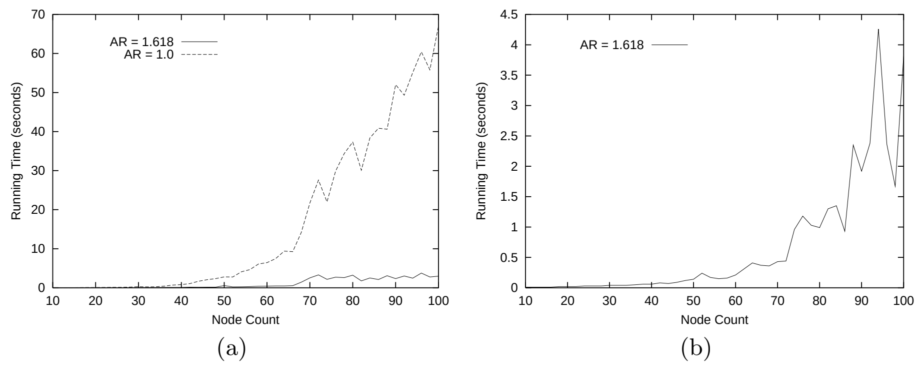


**Fig. 8.** Running time of `ULair` for (a) the first group and (b) the second group of experiments.

## 7 Conclusions

The experimental results show that `ULair` combines the positive attributes of the existing layering algorithms, namely, Longest Path, Coffman-Graham and Gansner. The width bound we specify when solving `ULair` represents the width of the final drawing much more accurately than the width bound used in the Coffman-Graham algorithm. The variety of parameters we can specify – width and height bounds, and the widths of nodes and edges (dummy nodes) – makes possible a family of layering solutions, permitting the user to choose the best one of them. This also makes `ULair` a good tool for studying from experimental point of view the relationship between various aesthetic parameters like the dimensions of a layering, the number of dummy nodes and the number of edge crossings in the final drawing.

As a final example we show two layerings of one of the small connected components of Graph A from the Graph Drawing Contest 2000 [2]. Graph A represents the structure of a software system from programmer's point view and contains large node labels. The layering in Figure 9 is a result of Gansner and the layering in Figure 10 is one of the alternative solutions given by a `ULair` when we draw the DAG putting different upper bounds on its width.

Our further work will be a more detailed study of the structure of the layering polytope $\mathcal{GLP}(\mathcal{L}_G^H, W)$ in terms of valid inequalities and facets in order to develop a branch-and-cut algorithm for solving `ULair`, and comparing its performance to the present solution method. This, we believe, will make it possible to accurately layer graphs well in excess of the 100-node examples that we have been working with.

**Acknowledgments.** We would like to thank Petra Mutzel and Gunnar Klau for the graph database they kindly let us use for performing our experiments.

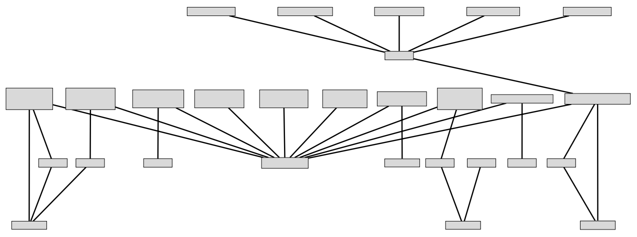


**Fig. 9.** Graph A: Gansner's layering (all edges point downwards).

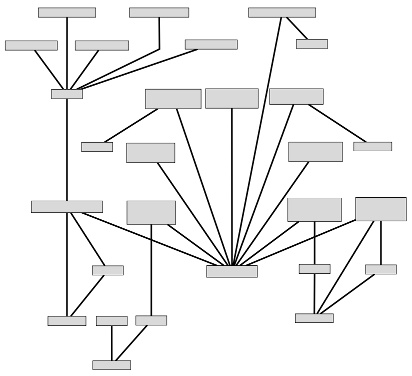


**Fig. 10.** Graph A: `ULair`'s layering (all edges point downwards).

## References

1. G. Di Battista, A. Garg, G. Liotta, R. Tamassia, E. Tassinari, and F. Vargiu. An experimental comparison of four graph drawing algorithms. *Computational Geometry: Theory and Applications*, (7):303 – 316, 1997.
2. Fr. Brandenburg, Ul. Brandes, M. Himsolt, and M. Raitner. Graph-drawing contest report. In Joe Marks, editor, *Graph Drawing: Proceedings of 8th International Symposium, GD 2000*, pages 410 – 418. Springer-Verlag, 2000.
3. E. G. Coffman and R. L. Graham. Optimal scheduling for two processor systems. *Acta Informatica*, 1:200–213, 1972.
4. P. Eades and K. Sugiyama. How to draw a directed graph. *Journal of Information Processing*, 13(4):424–437, 1990.
5. E. Gansner, E. Koutsofios, S. North, and K. Vo. A technique for drawing directed graphs. *IEEE Transactions on Software Engineering*, 19(3):214–229, March 1993.
6. K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierarchical system structures. *IEEE Transaction on Systems, Man, and Cybernetics*, SMC-11(2):109–125, February 1981.