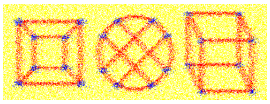


Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 15, no. 1, pp. 127–155 (2011)

Upward Planarization Layout

Markus Chimani[1] *Carsten Gutwenger*[2] *Petra Mutzel*[2] *Hoi-Ming Wong*[2]

[1] Algorithm Engineering Group,
Friedrich-Schiller-University Jena, Germany
[2] Chair for Algorithm Engineering,
TU Dortmund, Germany

Abstract

Recently, we presented a new practical method for upward crossing minimization [8], which clearly outperformed existing approaches for drawing hierarchical graphs in that respect. The outcome of this method is an *upward planar representation (UPR)*, a planarly embedded graph in which crossings are represented by dummy vertices. However, straight-forward approaches for drawing such UPRs lead to quite unsatisfactory results. In this paper, we present a new algorithm for drawing UPRs that greatly improves the layout quality, leading to good hierarchal drawings with few crossings. We analyze its performance on well-known benchmark graphs and compare it with alternative approaches.

Submitted:	Reviewed:	Revised:	Accepted:	Final:
December 2009	August 2010	August 2010	November 2010	November 2010
		Published: February 2011		
	Article type: Regular paper		Communicated by: D. Eppstein and E. R. Gansner	

Markus Chimani is funded by a Carl-Zeiss-Foundation juniorprofessorship.
Hoi-Ming Wong was supported by the German Research Foundation (DFG), priority project (SPP) 1307 "Algorithm Engineering", subproject "Planarization Practices in Automatic Graph Drawing".
E-mail addresses: markus.chimani@uni-jena.de (Markus Chimani) carsten.gutwenger@cs.tu-dortmund.de (Carsten Gutwenger) petra.mutzel@cs.tu-dortmund.de (Petra Mutzel) hoi-ming.wong@cs.tu-dortmund.de (Hoi-Ming Wong)

1 Introduction

The visualization of hierarchical graphs representing some natural flow of information is one of the key topics in graph drawing. It has numerous practical applications and has received a lot of scientific attention since the very beginning of graph drawing. Formally, we are given a directed acyclic graph (DAG) G and we want to find an *upward drawing* of G, i.e., a drawing of G in which all arcs are drawn as curves monotonically increasing in the vertical direction.

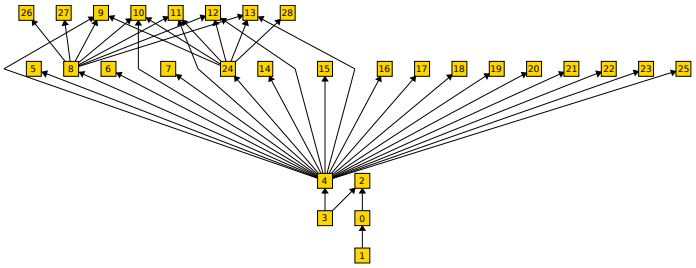
In 1981, Sugiyama et al. [20] proposed their well-known three-phase framework for creating such drawings, which is still widely used:

1. **Layer assignment:** Assign nodes to layers such that arcs point from lower to higher layers; split long arcs spanning several layers by creating *dummy nodes*.
2. **Node Ordering/Crossing reduction:** Order nodes on the layer to reduce the number of arc crossings.
3. **Coordinate assignment:** Assign coordinates to original nodes and dummy nodes (bend points) such that we get few bend points and short arcs.

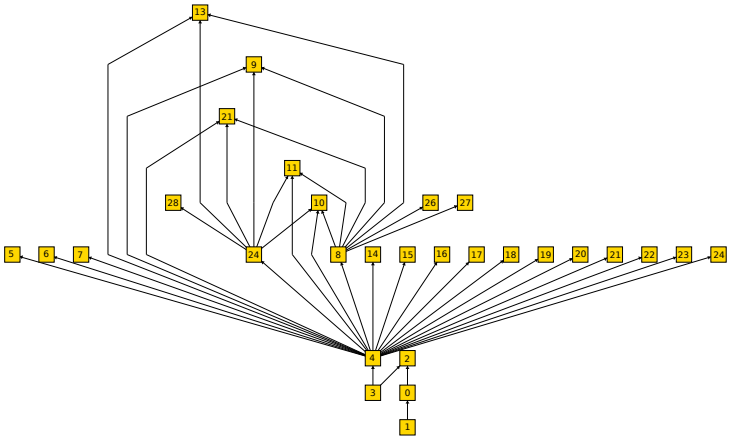
A vast number of modifications and alternatives for the individual steps have been proposed; e.g., Gansner et al. [13] give an LP-based formulation for layer and coordinate assignment. Their layer assignment computes a layering which minimizes the sum of the vertical edge lengths (i.e., the number of layers an edge spans). Their coordinate assignment minimizes the objective function $\sum_{e=(u,v)\in A} w(e) \cdot |X(u) - X(v)|$, where $w(e)$ gives the priority for drawing e vertically and A is the arc set after splitting long arcs. Brandes and Köpf [4] propose an approach which is simpler and faster, but nevertheless it computes coordinate assignments with similar quality. Branke et al. [5] investigate the computational complexity of the *width-restricted graph layering problem*. They prove that width-restricted graph layering is NP-hard when taking the dummy nodes into account. Healy and Nikolov give an experimental analysis of existing layering algorithms for DAGs [16]. They also present an ILP formulation and a branch-and-cut algorithm for layering a DAG with a minimum number of dummy nodes, where in addition, upper bounds for the width and height of the layering are given [15].

However, a major drawback of Sugiyama's framework could not be solved by any of these modifications: Since layer assignment and crossing reduction are realized as independent steps, the resulting drawing might have many unnecessary crossings caused by an unfortunate layer assignment. A main challenge is to perform crossing reduction *without any* layer assignment. First steps to adapt the planarization approach for undirected graphs [1, 14] have been presented in [2, 9]; Eiglsperger et al. [12] presented the more advanced *mixed upward planarization* approach. However, even the latter approach still needs some kind of layering. Experimental results suggested that this approach produces considerably fewer crossings than Sugiyama's algorithm. Previously [8], we presented a novel approach for upward planarization that does not require any layering. We could experimentally show that this new approach clearly outperforms Eiglsperger's mixed upward planarization and Sugiyama's algorithm with respect to crossing reduction.

The output of an upward planarization procedure is an *upward planar representation*, i.e., a representation of the original digraph, in which crossings are replaced



(a) Sugiyama (24 crossings)



(b) Upward planarization (4 crossings)

Figure 1: Instance *g.29.16* (North DAGs) with 29 nodes and 38 arcs.

by dummy vertices (*crossing dummies*) and a planar embedding with designated external face is given. In our case, the upward planar representation will always be a single-source, single sink embedded digraph; if the input digraph contains multiple sources we introduce a super-source $\hat{s}$ connected to all sources and do not count crossings with arcs incident to $\hat{s}$.

A simple method to draw a DAG by applying upward planarization consists of using Sugiyama's coordinate assignment phase for drawing the upward planar representation, where we use a straight-forwardly obtained layering and the ordering of the nodes on each layer implied by the upward-planar representation and embedding. However, this method (denoted by *UPSugiyama* in the following) produces quite unsatisfactory drawings with too many layers and much too long arcs. The main objective of this paper is to significantly improve on this simple method, by enhancing the computation of layers and node orderings and taking the special roles of crossing dummies into account. This will allow us to reduce the heights of the drawings and lengths of the arcs substantially, resulting in much more pleasant drawings.

Since upward planarization yields an upward planar representation, we can alternatively use drawing methods for upward planar digraphs to draw it, cf. [9]. We consider two such algorithms in our experimental study:

Dominance drawings: The linear-time algorithm by Di Battista et al. [11] draws

planar *st*-digraphs with small area on a grid. We apply this algorithm by augmenting the upward planar representation to a planar *st*-digraph and omitting the augmenting arcs in the final drawing.

Visibility representations: We use the algorithm by Rosenstiehl and Tarjan [19] for computing a visibility representation on the grid. This algorithm is based on bipolar orientations implied by an *st*-numbering. By augmenting the upward planar representation to an *st*-planar digraph, we obtain a bipolar orientation such that the resulting drawing is upward planar. Again, we omit augmenting edges in the final drawing.

Fig. 1 shows a relatively small digraph, where the benefits of the new upward planarization approach can be easily seen: While the classical Sugiyama approach leads to few layers, our approach can expand the layout of the subgraph that looks very congested otherwise.

Overview of the Frameworks. We give here an overview of the drawing algorithms we encountered in our experiments (cf. Fig. 2). They are: the classical Sugiyama framework (*Sugiyama*), our new upward planarization layout (*UPL*), the straight-forward approach (*UPSugiyama*), and the dominance and visibility drawing approaches (*Dominance* and *Visibility*). An example drawing of each approach is given in Fig. 3.

When considering the layer assignment as a sub-step of coordinate assignment, the drawing frameworks based on upward planarization can be properly divided into two main phases: The crossing minimization and the coordinate assignment phase. The coordinate assignment phase of *UPL* and *UPSugiyama* can further be divided into two sub-steps: the layering/node ordering step and final coordinate assignment.

Upward Planarization. We briefly sketch the upward planarization approach proposed in [8]. It can be divided into two phases: the feasible subgraph computation and the reinsertion phase. In the first phase the input DAG G is transformed into a single-source digraph G' by adding an artificial super source $\hat{s}$ and connecting it to the sources of G. Then we compute a spanning tree T of G' and iteratively try to insert the remaining arcs into T. Thereby, we perform a subgraph feasibility test after each inserted arc e: we not only test upward planarity but also if all remaining edges can potentially still be inserted (with crossings) in an upward fashion. If the resulting digraph is not feasible in this sense, we add e to a set of deleted arcs B instead. By applying these operations, we obtain an embedded feasible upward planar subgraph U.

In the second phase, the arcs in B are reinserted into U one after another such that few crossings arise. Thereby, the crossings caused by the reinsertion are replaced by crossing dummies. As a result, we obtain an upward planar representation of G'. This representation is a single-source, but usually not a single-sink digraph. By adding additional arcs (*sink-arcs*) for each internal face using a so called *face-sink graph* [3] and by adding an additional super sink $\hat{t}$ that is connected with the former sinks on the external face (cf.Fig. 4), the representation becomes an upward embedded single-source, single-sink digraph R.

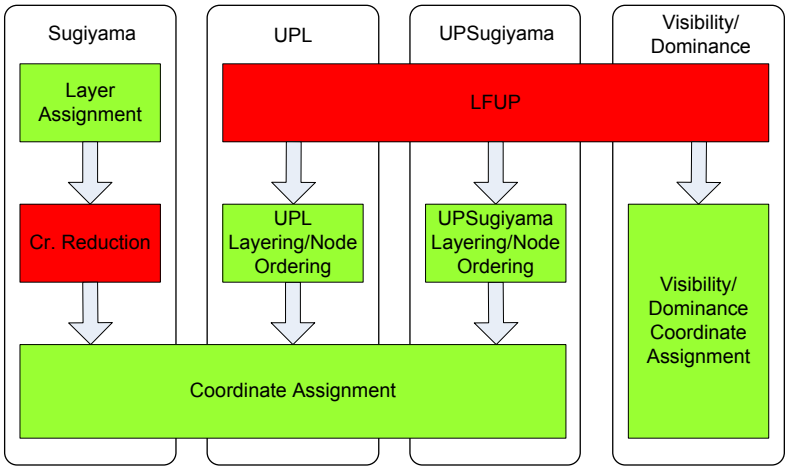


Figure 2: Overview of the frameworks: the classical drawing framework by Sugiyama et al. (*Sugiyama*); our new upward planarization layout approach (*UPL*); the straightforward application of the Sugiyama framework on the upward planar representation (*UPSugiyama*); the dominance and visibility approach (*Dominance* and *Visibility*). For the latter three drawing algorithms we use the layer free upward crossing minimization approach (*LFUP*) by Chimani et al. [8] for obtaining an upward planar representation.

The runtime of the first phase is $\mathcal{O}(|A|^2)$ and the runtime for the second phase $\mathcal{O}(|A|^5)$. Therefore, an upward planarized representation for any connected DAG $G = (V, A)$ can be computed in time $\mathcal{O}(|A|^5)$.

In the following, an upward planar representation is always an augmented embedded graph R. Let v and e be a node and an arc in G, respectively. We denote the corresponding node and arc in R by v_R and e_R, respectively.

Organization of the Paper. In Sect. 2, we show how to perform layer, node order, and coordinate assignment using an upward planar representation, as well as some further beautifications. In Sect. 3, we experimentally evaluate our algorithm and compare it with existing approaches and in Sect. 4 we give a conclusion on the new drawing method. Sect. 5 shows some example drawings, comparing the results obtained by our approach with drawings produced by applying the classical Sugiyama framework.

2 Upward Planarization Layout Algorithm

Let R be an upward planar representation of a DAG G. Let $\mathcal{D}^*$ be an upward drawing induced by R such that all crossing dummy nodes are replaced by crossings and all auxiliary arcs and nodes are omitted. A drawing $\mathcal{D}$ is a *realization* of R if for each node v of G, the arc order of v induced by $\mathcal{D}$ coincides with the arc order induced by $\mathcal{D}^*$ and the crossings arising in $\mathcal{D}$ are the ones modeled by R.

The crucial starting point of our algorithm can be stated as follows:

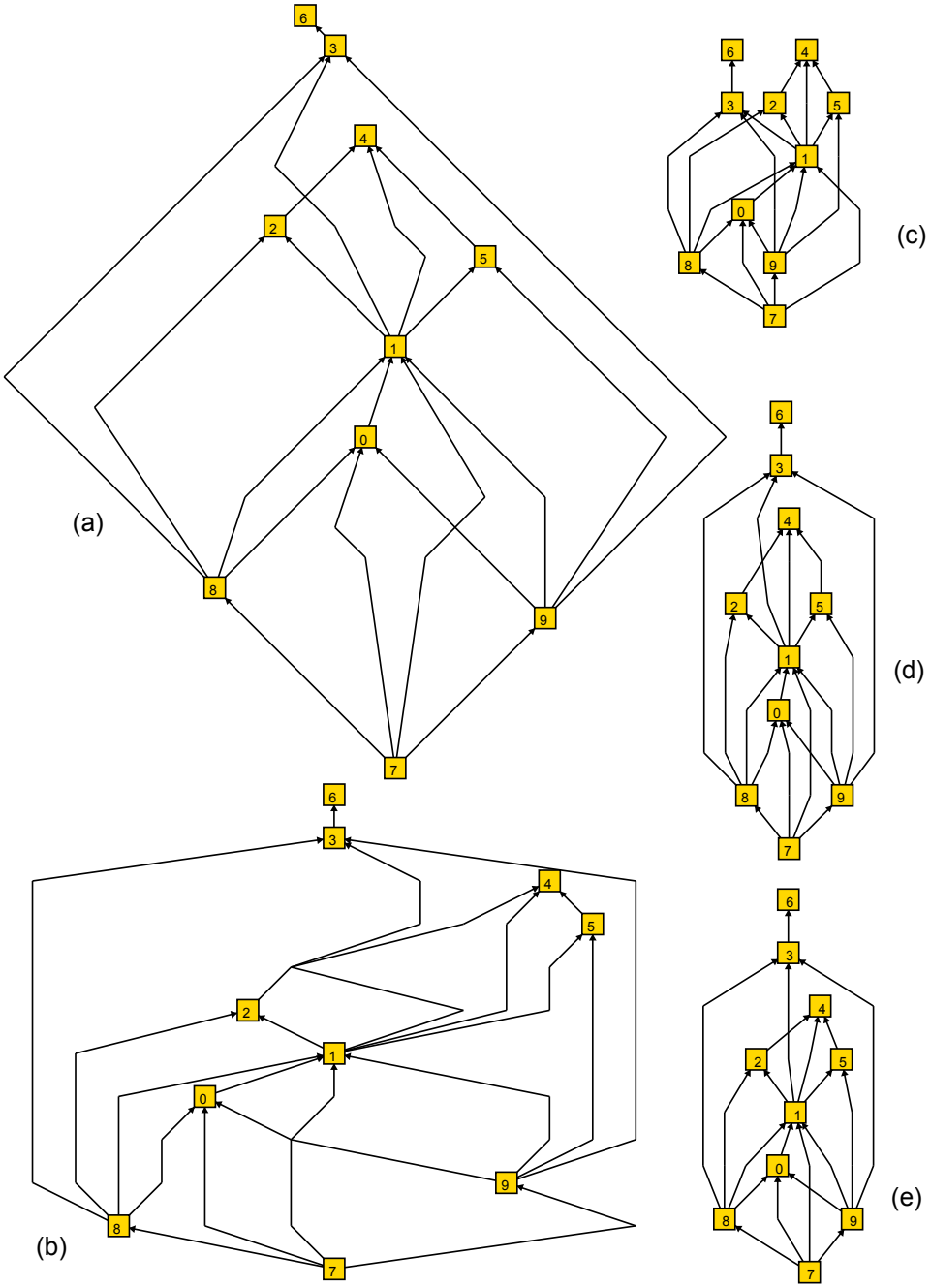
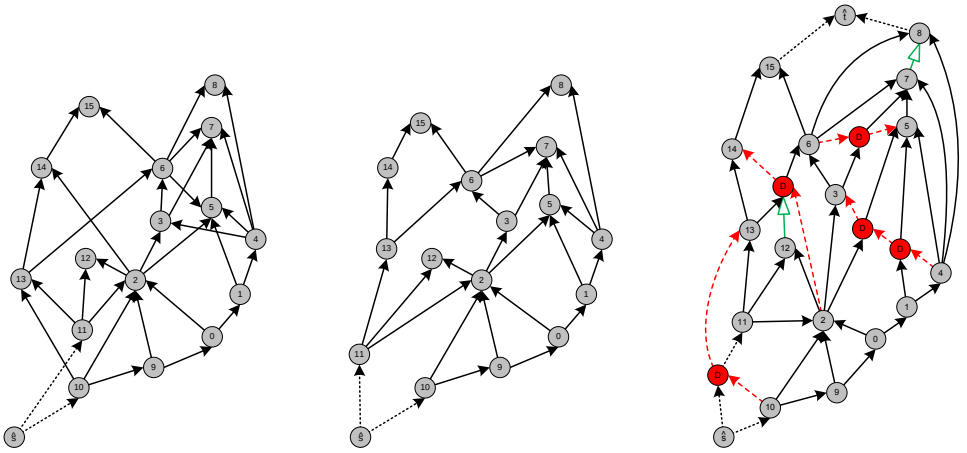


Figure 3: Layout of instance *g.10.19* (North DAGs): (a) Dominance; (b) Visibility; (c) Sugiyama; (d) UPSugiyama; (e) UPL; the drawings (a), (b), (d) and (e) are based on a common upward planar representation produced by [8].



(a) (b) (c)

Figure 4: Illustration of the upward planarization approach by Chimani et al. [8]: (a) input DAG G augmented to G' via the artificial super source $\hat{s}$; (b) embedded feasible subgraph U of G' obtained by deleting the arcs $(10, 13), (2, 14), (4, 3), (6, 5)$; (c) upward planar representation R of G after reinserting the deleted arcs (dashed line). R contains five crossing dummies. By adding sink-arcs (drawn with hollow arrow heads) and the super sink $\hat{t}$, R becomes a single-source, single-sink digraph.

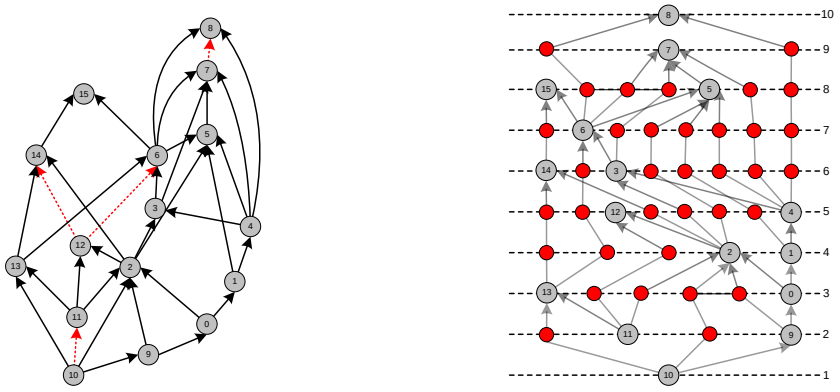
Proposition 1 *Given an upward planar representation R of a DAG G, there exists a layering of the nodes of G, a node order per layer, and a node placement including bend points, such that the thereby induced drawing of G realizes R.*

We observe that a realizing drawing of G hence follows Sugiyama's framework, but the individual steps do not simply optimize their respective objectives but follow the overall goal of simulating R. Our algorithm hence divides naturally into the three steps known from Sugiyama's framework, whereby the first two steps are closely related.

As sketched in Sect. 1 (and investigated in the experimental comparison; see Sect. 3.1), it is easy to find *some* solution that realizes R. Yet, even if G is only of moderate size, R can become much larger due to the number of inserted dummy nodes. This causes weak runtime performance, many layers, and overall unsatisfactory drawings. Hence our algorithm aims at minimizing the required layers, thereby also reducing the necessary dummy nodes resulting from splitting long arcs.

2.1 Layer Assignment and Node Ordering

Layer Assignment. Let H be a copy of G that we will use to obtain a valid layering for G; cf. Fig. 5. For any two nodes $u, v \in V(G)$, we add an auxiliary arc (u, v) to H, i.e., $H = H + (u, v)$ if: (a) there exists no directed path from u to v in G and in H, but (b) there exists a directed path from u_R to v_R in R. Part (a) prohibits the unnecessary generation of transitive arcs; part (b), in conjunction with the sink-arcs and the single-source, single-sink property of R, ensures that the hierarchical order



(a) The auxiliary digraph H with respect to R from Fig. 4. Arcs in H but not in G are drawn as dotted lines.

(b) Layering of H. Long-arc dummies are drawn as smaller circles.

Figure 5: Layer Assignment and Node Ordering

of R is mapped to H. Since G and R are DAGs, H is also acyclic and we can use any existing layering algorithm on H. Let $\mathcal{L} = \langle L_1, L_2, ..., L_\ell \rangle$ be a layering for H, and therefore also for G; i.e., $\dot{\bigcup}_{1 \le i \le \ell} L_i = V(G)$. We can extend this layering naturally, by splitting any arc that spans more than one layer into a chain of arc segments by introducing dummy nodes (*long-arc dummies*).

Node Ordering. Considering the layering $\mathcal{L}$, we now have to arrange the nodes on each layer according to the order induced by R. For this purpose, we consider the order of the arcs around each node, as given by R. In particular, we can recognize the *left incoming* arc for any node v, which is the embedding-wise left-most arc with target v. Note that this arc is defined for each node except for the super source.

Now consider any two distinct nodes u and v on the same layer. We can decide their correct order using the following strategy: we construct a *left path* p_u from $\hat{s}$ to u_R from back to front, i.e., starting at u_R we select its left incoming arc e as the end of p_u and proceed from the source node of e, choosing its left incoming arc as the second to last arc in p_u, and so on. The construction of p_u ends when we reach the super source, which will always happen as R is a single-sink DAG. Analogously, we construct the left path p_v from $\hat{s}$ to v_R.

The paths p_u and p_v may share a common subpath starting at $\hat{s}$; let c_R be the last common node of p_u and p_v, and let e_u and e_v be the first different arcs, respectively. We determine the ordering of u and v directly by the order of e_u and e_v at c_R. For example in Fig. 5, if u and v are the nodes '4' and '12' respectively (layer 5), then node '$\hat{s}$' is the last common node of their corresponding left paths in R (see Fig. 4(c)), and hence v is left of u.

Algorithmically, we can consider each layer independently. Introducing an auxiliary digraph A, the above relationship between two nodes on the same layer can be modeled as an arc between these two nodes in A. We can construct a correct ordering for the layer by computing the topological order in A. Note that therefore we do not have to compute the arc direction for all node pairs, but only for the ones that are

not already "solved" by other arcs through transitivity. We obtain a final layering realizing R.

The height of a layering $\mathcal{L}$ is the number of layers of $\mathcal{L}$. Our layering algorithm above has the following property:

Lemma 1 *Let R be an upward planar representation of a single-source DAG G and $\mathcal{L}$ a layering realizing R such that $\mathcal{L}$ is obtained by applying a longest path algorithm on H. Let h be the height of $\mathcal{L}$. Then for any layering $\mathcal{L}'$ with height h', $h \leq h'$ holds.*

Proof:

Let p be a directed path from u_R to v_R in R such that there is no corresponding directed path from u to v in G and H (condition (a)). For convenience we identify p with a sequence of arcs $e_1, \ldots, e_k$. Notice that an arc e_i of p cannot be adjacent to the super sink $\hat{t}$ of R, since $\hat{t}$ is not a node of G. Also, e_i is not associated with an arc added for connecting the super source with the sources of G, since G is a single-source digraph.

We prove by induction over k that adding an auxiliary arc $a = (u, v)$ to H is necessary in order to ensure the hierarchy order between node u and v. In particular, a cannot be omitted if the layering $\mathcal{L}$ shall realize R.

Induction basis ($k = 1$): p contains only one arc $e_1 = (u_R, v_R)$. Since u_R and v_R are not crossing dummies, e_1 is not an arc segment of any arc. (An arc segment arises by splitting an arc during the upward planarization.) Also, e_1 does not correspond to a "real" arc of G due to condition (a); thus e_1 is a sink-arc. u is a sink (*sink-switch*) and v is the topmost node (*top-sink-switch*) of an inner face f of an upward planar drawing of R. Obviously, v must be drawn higher than u in a realization of R. Therefore adding arc a to H ensures the hierarchy order between u and v.

Induction step: We assume that the hierarchy order of the nodes of $p' = e_1, \ldots, e_j$ with $j < k$ is mapped to H by adding some auxiliary arcs to it.

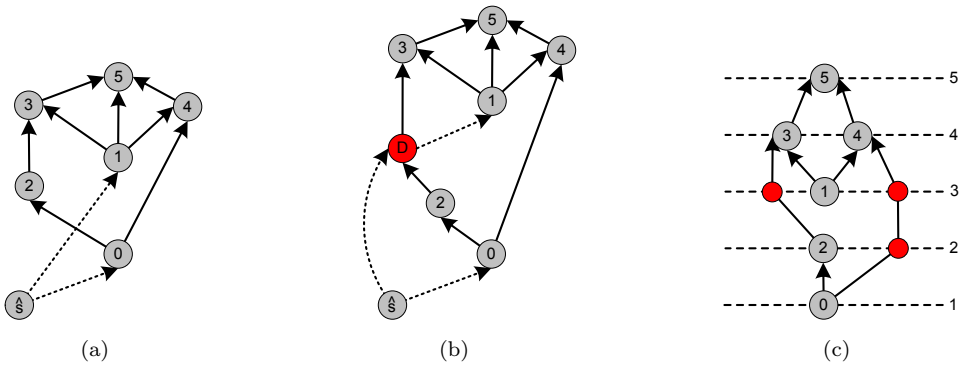
Let $p = e_1, \ldots, e_k$ with $e_k = (x_R, v_R)$. There are three possible cases:

(i) e_k is a sink-arc: The fact that x is layered higher than u is already mapped to H (induction assumption). Since e_k is a sink-arc, v must be layered higher than x (see induction basis), and therefore higher than u. Adding arc a to H ensures the hierarchy order.

(ii) There is a "real" arc (x, v) in G corresponding to e_k: By the induction assumption the hierarchy order between u and x is mapped to H by adding some auxiliary arcs. Hence there is a path from u to x in H and, since H is a copy of G, there is also a path from u to v. Due to condition (a) no auxiliary arcs are added to H.

(iii) e_k is an arc segment: x_R is a crossing dummy and the target node of arc e_{k-1}. We have two sub-cases:

Case e_{k-1} and e_k are arc segments of a common arc $b = (w, v), b \in G$: By the induction assumption, the hierarchy order between node u and w is mapped to H. Since w must be layered higher than u and since the arc $b = (w, v)$ exists, v must be layered higher than w and therefore higher than u. Adding arc a to H ensures this fact.



(a) (b) (c)

Figure 6: An example of a layering with unnecessary layers: (a) input DAG G which is augmented to an sT-graph; (b) an upward planar representation R of G; (c) a layering of G realizing R obtained by our layering and node ordering approach. The number of layers can be reduced by one if we assign node '1' to layer 2 (as the right neighbor of node '2'), node '3' and '4' to layer 3, and node '5' to layer 4.

> **Case** e_{k-1} and e_k are not arc segments of a common arc: Let e_{k-1} be an arc segment of arc $d = (w, \cdot)$ and e_k be an arc segment of arc $b = (\cdot, v)$. The crossing dummy x_R models the crossing ξ between arc b and arc d. Therefore ξ must be drawn higher than w in an upward drawing $\mathcal{D}$ and v must be drawn higher than ξ; otherwise the line segment from ξ to v in $\mathcal{D}$ would not point monotonically increasing in the vertical direction. Thus v must be layered higher than u. We have to add the arc a to H to reflect this fact.

The induction proof reveals that no additional auxiliary arcs of H can be omitted; otherwise the hierarchy order of the nodes may not be mapped to H correctly. Hence a layering $\mathcal{L}$ with respect to R obtained by applying a longest path algorithm to H contains no unnecessary layers. After computing the node order and rearranging the nodes of $\mathcal{L}$, $\mathcal{L}$ is a realization of $\mathcal{R}$.

□

Unfortunately, our layer assignment approach can compute a layering with unnecessary layers for DAGs with multiple sources. Fig. 6 gives an example for such a case. The path from node '2' to node '1' arises in the representation due to the arc $(\hat{s}, 1)$. Therefore, node '1' is layered higher than actually necessary.

2.1.1 Long-Arc Dummy Reduction

A *dominated subgraph* of G w.r.t. a node s is the subgraph induced by the nodes v for which G contains a directed path from s to v. Most layering algorithms—in particular also the optimal LP-based approach [13]—will put the nodes on the lowest possible layer. While this is generally a good idea, this approach can be counter-productive in the context of the super source node that will be removed from the final drawing: Since every source node s in G is attached to the super source node $\hat{s}$ (which is on the lowest layer), s may end up very low in the drawing, even though most of its dominated subgraph requires higher layers, hence introducing long arcs.

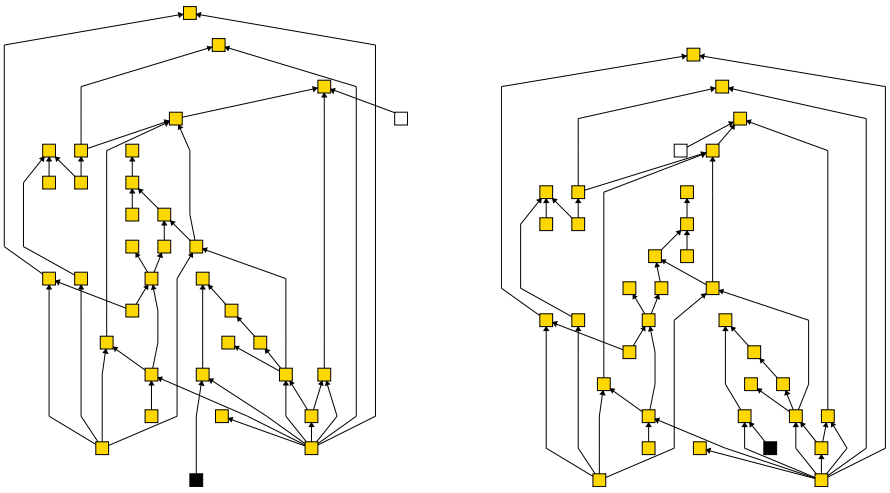


Figure 7: A drawing of graph *grafo2379.35* (Rome graphs): (left) without postprocessing, (right) after applying source repositioning (white node) and long-arc dummy reduction (black node).

We tackle this problem using an approach similar to the *promotion node method* by Nikolov and Tarassov [17] by re-layering parts of the dominated subgraphs after the removal of $\hat{s}$, without modifying the hierarchical order induced by R. Layers that become empty by these operations can be removed afterwards:

For Each source s in G (in decreasing order of their layer index j):

(a) Mark the subgraph dominated by s. Let M_i be the marked nodes on layer L_i $(1 \leq i \leq \ell)$.

(b) **For** $i = j + 1$ **To** ℓ:

If M_i are all long-arc dummies **Then**

(i) Remove the nodes M_i and lift the marked subgraph on the layers below L_i by one layer.

(ii) **If** the new layering causes more edge crossings **Or** more long-arc dummies **Then** Undo step (i) and **Break** (continue with next source)

2.1.2 Repositioning the Sources

Since our upward planarization algorithm considers G augmented with $\hat{s}$ and inserts additional arcs considering a fixed embedding, the final upward planar representation may contain artifacts in the form of seemingly unnecessary crossings; see, e.g., the white node in Fig. 7. To overcome this, we sift each source s through all possible positions on its layer and choose the position where it causes the fewest crossings.

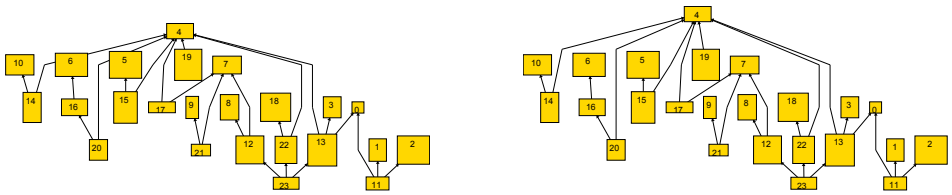


Figure 8: A drawing of graph *grafo159.24* (Rome graphs) with random node sizes: without (left) and with (right) our bending arcs method and individual layer distance assignment.

2.2 Coordinate Assignment

After the previous steps we get a correct layering and node ordering realizing R. Conceptually, we can use *any* coordinate assignment strategy (e.g., [6, 13]) known for Sugiyama's layout algorithm; it will always preserve the given number of crossings. All these methods assign horizontal coordinates to the nodes while preserving the given node ordering on each layer. The aim is to generate drawings such that the subdivided long arcs are drawn as vertical straight lines for their most part.

Yet, when considering the hard-to-measure "beauty" or "readability" of the resulting drawings, we realize that we can improve on traditional coordinate assignment strategies as they usually do not accommodate the following two drawing problems:

- *node-arc crossings*: A line segment connecting nodes or bend points between layer L_i and L_{i+1} may cross through some nodes of these two layers. This can easily happen when node sizes are relatively large compared to the layer distance.

- *long-line segments*: The general direction of upward drawings should naturally be along the vertical direction. Yet, there can be arc segments between some layers L_i and L_{i+1} which are very long since they span a large horizontal distance. Such arcs can make Sugiyama-style drawings hard to read.

Fig. 8 shows the benefit of the two strategies described below. Note that these strategies are not only applicable to our layout algorithm, but to any Sugiyama-style layout.

2.2.1 Vertical Coordinates

Usually, the vertical coordinates for the nodes on layer L_i are simply given by $\delta \cdot i$, where δ is the minimal layer distance. Yet, often we may prefer larger distances between layers in order to improve readability: larger distances counter both above problems, but in our context we are in particular interested in long-line segments—we will discuss how to tackle node-arc crossings in Sect. 2.2.2.

Buchheim et al. [6] propose a solution for variable layer distances depending on the gradient of the line segments. However, our experimental results show that drawing DAGs using upward planarization tends to produce drawings with large height. Therefore we use a different approach which limits the maximal layer distance to 3δ.

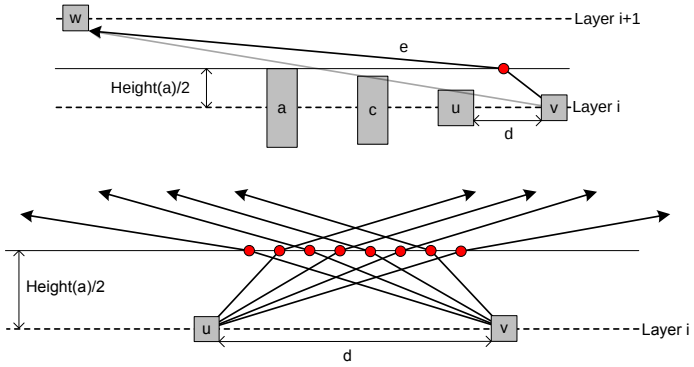


Figure 9: Avoiding node–line overlapping by introducing new bend points into an arc e (top); horizontal coordinates of bend points must all be distinct, even if all involved arcs require a bend (bottom).

Let σ_i be the number of arcs between L_i and L_{i+1} whose lengths are at least 3δ. We set the vertical distance between these two layers to $(1 + \min\{\sigma_i/4, 2\})\delta$ (empirically evaluated).

2.2.2 Bending Arcs

While enlarging the layer distance also helps to prevent node–line crossings, the required increase in height is usually not worth it—from the readability perspective. We therefore propose a strategy that allows trading additional bend points for layer distance. The strategy can be parameterized to find one's favorite trade-off between these two measures, namely, increase the layer distances to reduce the number of bend points or keep the layer distances small, instead introduce new bend points to avoid node–line crossings.

Let $X(v)$ and $Y(v)$ denote the horizontal and vertical coordinates of a node v, respectively. An arc (or line segment) $e = (v, w)$ is pointing *upward from left to right* (*right to left*) if $X(v) < X(w)$ ($X(v) > X(w)$, resp.). Since purely vertical line segments cannot cross through nodes, we distinguish four cases: e is pointing upward from right to left (or left to right) and v (or w) is a node on layer i. In all these cases, e has to bend if it overlaps some nodes of L_i. However, bending e might cause additional crossings. To avoid this, we also have to bend the line segments that cross the just bended line. W.l.o.g., we only discuss the case $X(v) > X(w)$ with $v \in L_i$. The other cases can be solved analogously.

Let width(v) and height(v) denote the width and height of the bounding box of a node v. Let a be the node on layer L_i with the highest bounding box, and let $\alpha :=$ height$(a)/2$. If v is a bend point and not shifted downwards before, then we do not need to introduce an additional bend. Instead we move v upwards by α. If v was already shifted downwards before due to one of our other cases, then we bend e by introducing a new bend point b and set $X(b) := X(v)$ and $Y(b) := Y(v) + 2\alpha$. We observe: By setting $Y(v)$ to α or in the latter case, setting the new bend point b to $Y(v) + 2\alpha$, we ensure that e cannot overlap any nodes on layer L_i.

Assume v is not a bend point. Then we have to introduce a bend point along

e. Thereby we have to consider that other arcs might also get rerouted and so we must accommodate enough space for them as well, such that no two bend points may coincide. In particular, it might be that the arcs leaving v's left neighbor to the right might also require additional bend points (cf. Fig. 9). Let u be the left neighbor of v on L_i and $d := X(v) - X(u) - \text{width}(v)/2 - \text{width}(u)/2$ their inner distance. Let r be the number of line segments adjacent to v and pointing from right to left; among these, assume that e is the j-th segment when counting from left to right. Let q be the number of line segments adjacent to u and pointing from left to right. Then, $\Delta := \frac{d}{q+r+1}$ gives the distances between the potential bend points, and the coordinates of the new bend point b are:

$$\begin{aligned} X(b) &:= X(u) + \frac{\text{width}(u)}{2} + \Delta \cdot (j + \min\{q, j-1\}) \\ Y(b) &:= Y(v) + \alpha \end{aligned}$$

In the worse case we have to introduce a new bend point for each line segment in order to prevent overlapping of the bend points and to prevent newly arising crossings (cf. Fig. 9 (bottom)). Therefore the number of newly introduced bend points for a layer L_i is bounded by the number of line segments connecting the nodes or bend points of L_i and L_{i+1}.

3 Experiments

We investigate the quality of our new algorithm in comparison with known algorithms. We first compare different approaches to draw a computed upward planar representation, i.e., if the crossing number is the most important factor in our drawing. Afterwards, we also compare our approach to Sugiyama's traditional framework.

All algorithms are implemented in the free and open-source (GPL) *Open Graph Drawing Framework (OGDF)* [18]. The experiments were conducted on an Intel Pentium 4 3.4Ghz PC with 2GB of RAM. All data points of the diagrams represent average values for the corresponding node or density group respectively. We use the following benchmark sets:

Rome Graphs: The Rome graphs [10] are a widely used benchmark set in graph drawing, which was obtained from a basic set of 112 real-world graphs. It contains 11528 instances with 10–100 nodes and 9–158 edges. Although the graphs are originally undirected, they have been used as directed graphs by artificially directing the edges according to the node order given in the input files [12, 8].

North DAGs: The North DAGs[1] have been introduced in an experimental comparison of algorithms for drawing DAGs [9]. The set consists of 1277 DAGs collected by Stephen North.

Since the North DAGs are a collection of heterogeneous digraphs, that is, the density of the digraphs with same number of nodes may vary from very dense to very sparse, we decided to group the DAGs into 9 sets, where the first set contains digraphs with 10 to 20 nodes and the i-th set contains digraphs with $10i + 1$ to $10(i + 1)$ nodes for $i = 2, \ldots, 9$.

[1] www.graphdrawing.org/data/index.html

Random DAGs: The real-world origin of the above benchmarks results in sets where, e.g., the relative *graph densities* (i.e., the ratio $|A|/|V|$ for a digraph $G = (V, A)$) are not uniformly distributed over the different graph sizes; in particular larger graphs tend to have lower densities. For a deeper investigation of our algorithm regarding graphs with high densities, we use a set of 200 random DAGs previously generated by us [7]. Each DAG has 100 nodes and each potential arc occurs with uniform probability p. The DAGs are grouped in 10 subsets such that each subset is generated with a certain p corresponding to the expected density $\varrho = |A|/|V| \in \{1.5, 2, 2.5, \ldots, 6\}$.

Although most real-world graphs have a density below 2–3, evaluating the drawing algorithms for these random DAGs can reveal some theoretical insight into the behavior of these algorithms.

3.1 Planarization Layouts

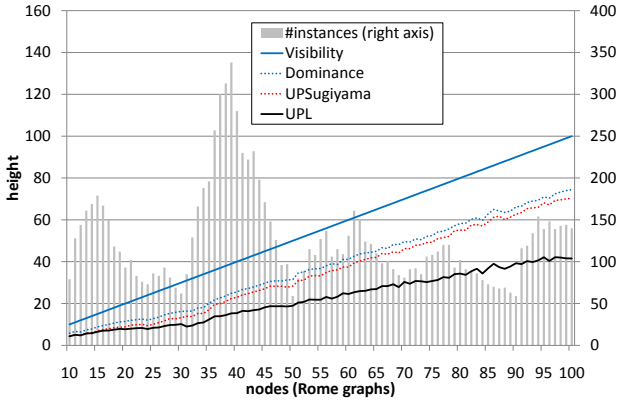
As outlined in the introduction, there are various other possibilities to draw an upward planar representation R of a digraph. Therefore, we use R also as input for alternative drawing algorithms. After computing the drawing, we can replace the dummy nodes by usual arc crossings and remove the sink-arcs and the super source/sink. By this approach we guarantee that the resulting drawing realizes the specified representation.

We compare the new drawing algorithm *UPL* to the dominance drawing style [11] (*Dominance*), the visibility representation drawing style [19] (*Visibility*), and a straightforward application of Sugiyama's framework (*UPSugiyama*). For the latter, we use an optimal ranking [13] for layering, extract the node orders directly from the upward planar representation, and apply the coordinate assignment algorithm by Buchheim et al. [6].

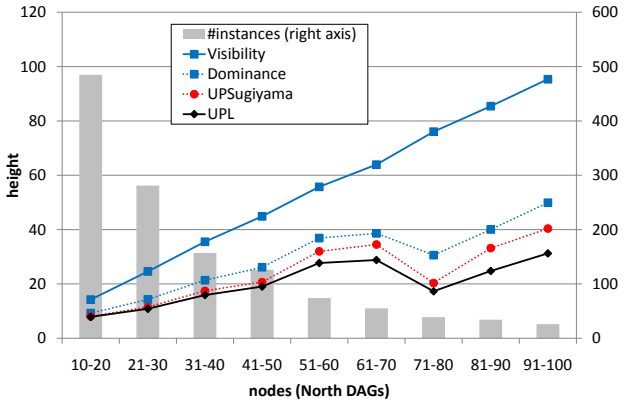
Fig. 10 and Fig. 11 show the average heights and widths of the resulting drawings for the Rome graphs, the North DAGs, and the random graphs, respectively. The average aspect ratios (i.e., the ratio width/height) of the drawings are given in Fig. 12. For a fair comparison between the different approaches, we want to disregard any differences which are only due to spacing parameters. Therefore, we use the following conventions: The *height* of a drawing is simply the number of required layers (in case of *UPL* and *UPSugiyama*) or the number of vertical grid coordinates (in case of *Dominance* and *Visibility*), respectively. The *width* of a drawing is the maximum number of *elements* per layer or horizontal grid line, respectively, where the elements on a layer or grid line ℓ are the nodes on ℓ as well as the edge lines crossing ℓ.

For a fair runtime comparison, we use the same coordinate assignment algorithm [6] for *UPL* and *UPSugiyama*. This choice is due to the fact that the alternative LP-based approach [13] requires too much time for *UPSugiyama*, because it has to consider very large digraphs due to the crossing and long-arc dummies. Note that the height and width measures defined above are invariant under the choice of the coordinate assignment algorithm. The average runtimes (in seconds) are shown in Fig. 13. We omit showing runtimes for the linear-time algorithms *Dominance* and *Visibility*, since they are usually below any measurable threshold. Instead, we give their average and maximum runtime values in Table 1.

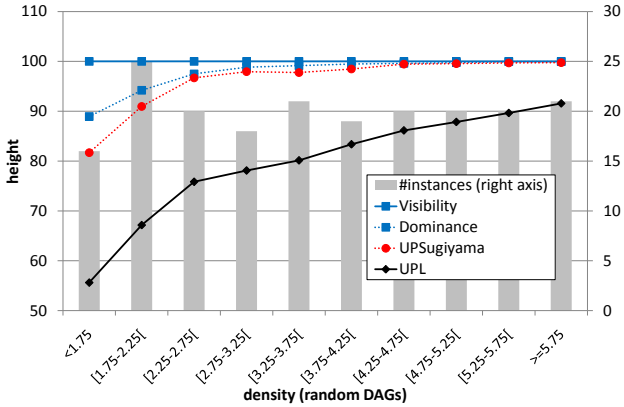
We observe that our new approach clearly outperforms all other approaches with respect to drawing height. Except for the North DAGs, the heights of the drawings increase with increasing number of nodes. This is not surprising, since with increasing



(a) Rome graphs

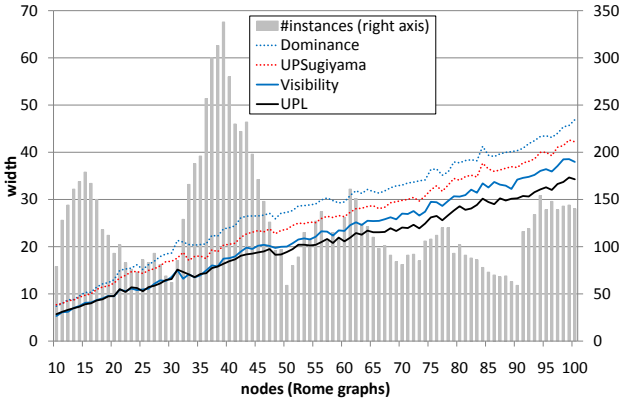


(b) North DAGs

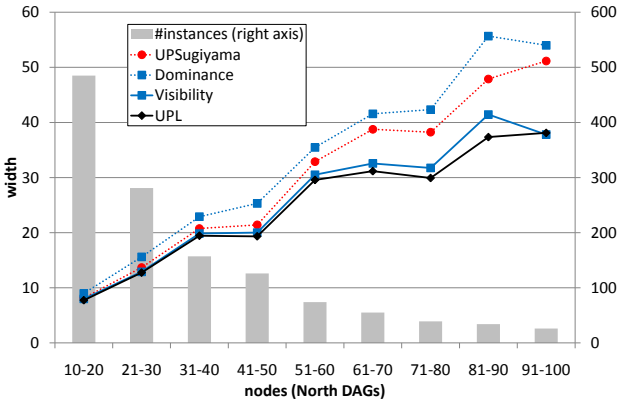


(c) random DAGs

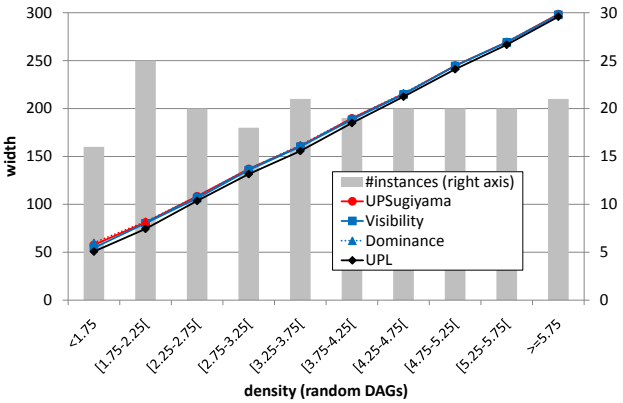
Figure 10: Drawing upward planar representations: average heights of the drawings.



(a) Rome graphs

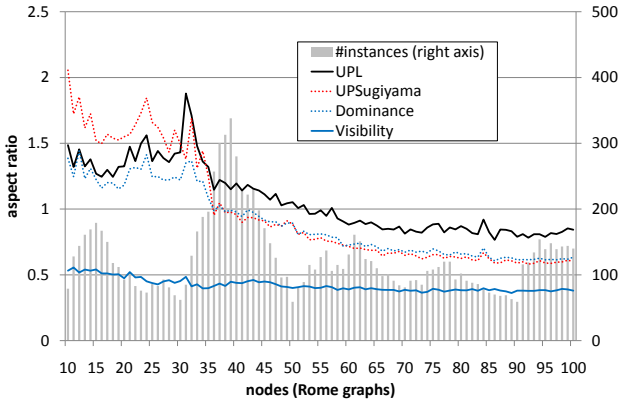


(b) North DAGs

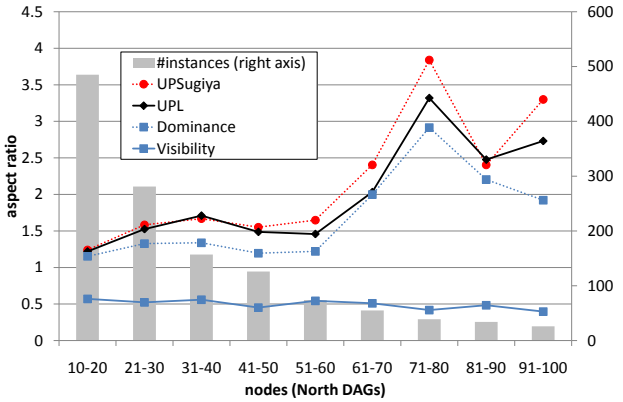


(c) random DAGs

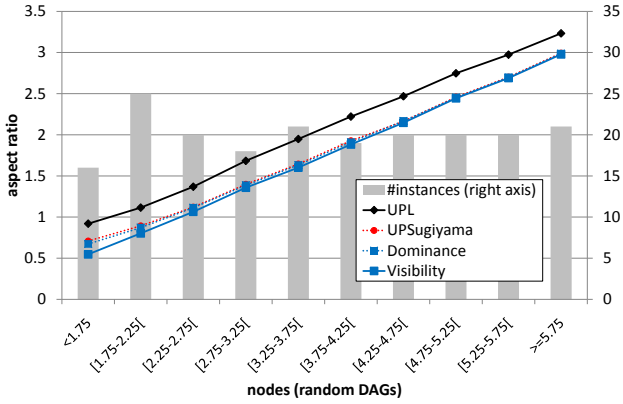
Figure 11: Drawing upward planar representations: average widths of the drawings.



(a) Rome graphs

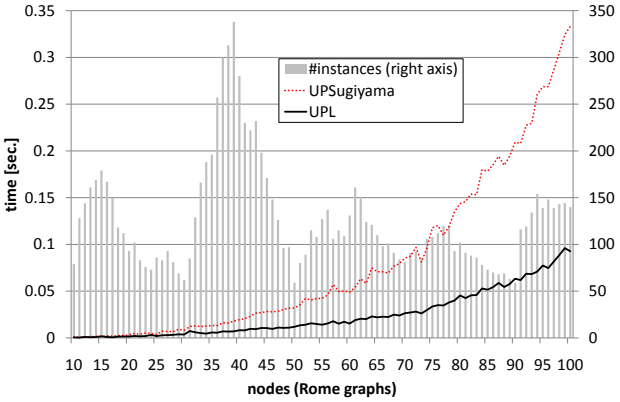


(b) North DAGs

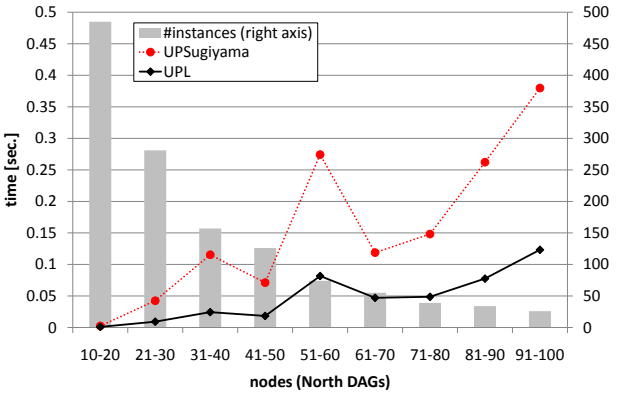


(c) random DAGs

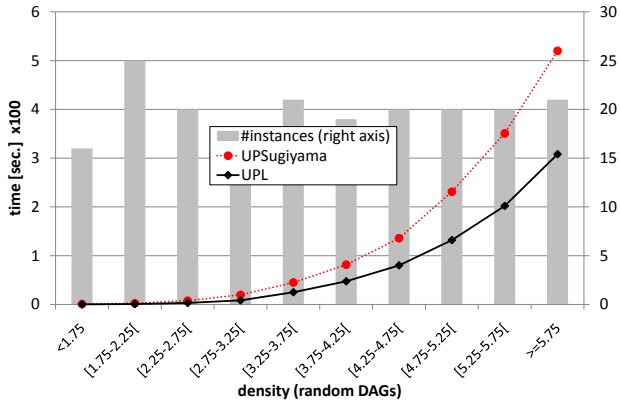
Figure 12: Drawing upward planar representations: average aspect ratios of the drawings.



(a) Rome graphs



(b) North DAGs



(c) random DAGs

Figure 13: Drawing upward planar representations: runtime.

Algorithm	Rome graphs		North DAGs		random DAGs	
	avg	max	avg	max	avg	max
Visibility	0.00067	0.016	0.00032	0.016	0.138	0.438
Dominance	0.00055	0.060	0.00039	0.032	0.122	0.390

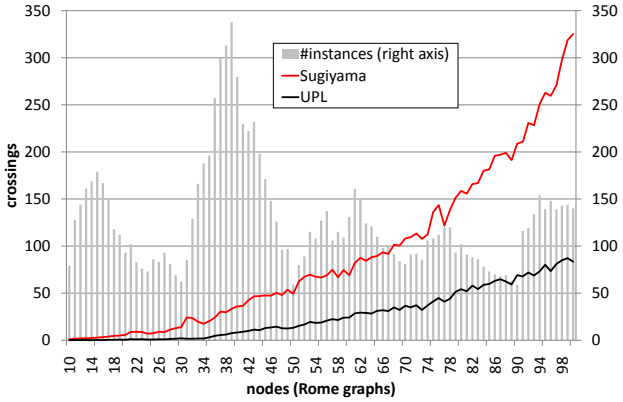
Table 1: Average and maximal runtime of *Visibility* and *Dominance* in seconds.

number of nodes the length of the longest path also increases. The diagram for the North DAGs (Fig. 10(b)) contains an unusual break at node group 71–80. The reason is that this group is the group with the shortest average longest path and the lowest average density. Furthermore, we observe that the heights of the drawings increase when the graphs becoming denser. Also the gap between *UPL* and the other planarization algorithms decreases; see Fig. 10(c). As the density increases, the height converges to the maximal possible height (see Fig. 10 (c)). Concerning drawing width *UPL* is clearly the winner on the Rome and North graphs benchmark sets, while all algorithms perform nearly the same on the random DAGs. As expected, the width increases when the instances becoming denser or when the number of nodes increases. This is due to the fact that digraphs with high numbers of nodes "need" more layers, and thus there are more arcs spanning more than one layer. Hence, the number of long arc dummies on a layer increases. If a digraph has a high density then more long arc dummies occur and the width also increases. For the same reason the aspect ratio of the random DAGs increase when the instances becoming denser. Regarding the runtime *UPL* is the clear winner compared to *UPSugiyama* (cf. Fig. 13).

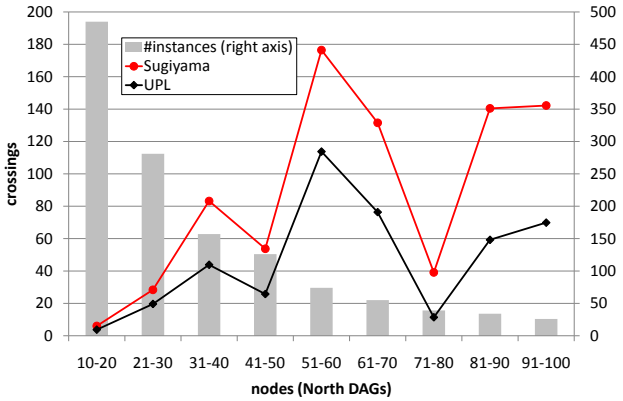
3.2 Comparison with Traditional Sugiyama

Finally, we can investigate how much the requirement of having a drawing with few crossings costs in terms of other quality measures. Therefore we compare *UPL* to a traditional Sugiyama approach that is not bound to a specific upward planar representation. For the latter we use the experimentally most competitive algorithms for the individual steps: We use the optimal LP-based approach for layering [13], the barycenter heuristic for the crossing reduction step (with best of 5 randomized runs), and assign the coordinates via the exact LP-based approach [13]. For a fair comparison, *UPL* also applies the LP-based coordinate assignment algorithm. This time, the runtime of *UPL* includes also the computation of the upward planar representation, because this step is not necessary for *Sugiyama*. We remark that the implementation of the planarization was vastly improved compared to previous results [8].

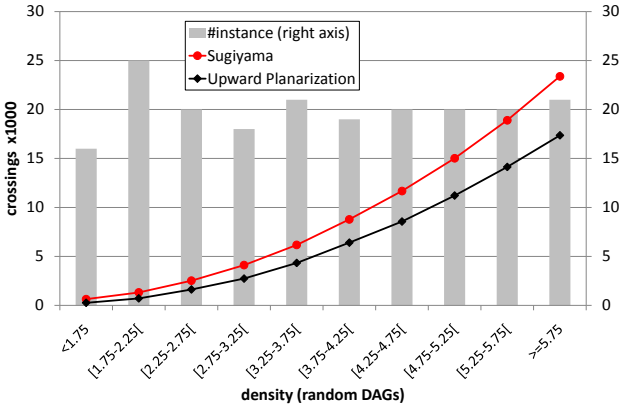
Clearly, the number of crossings in the pure Sugiyama approach is much higher (Fig. 14), which complies with the findings in [8]. As shown in Fig. 15, *UPL*'s drawings are of course higher than *Sugiyama*'s by construction. Like for the planarization layouts, the width and height of *Sugiyama* drawings increases when the number of nodes or the density of the graphs increases. While *UPL* produces often drawings with smaller widths for the Rome and North graphs (see Fig. 15 (a) and (b)), *Sugiyama* achieved better results for the random DAGs. This is due to the lower height of *Sugiyama* drawings. The Sugiyama approach requires fewer long arc dummies than *UPL*, which becomes more noticeable as the density is increasing. Yet we observe that the difference is not as large as expected, and *UPL* seems to be a good fit also



(a) Rome graphs

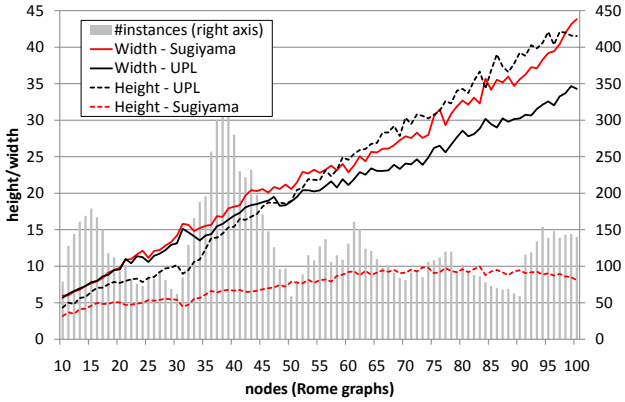


(b) North DAGs

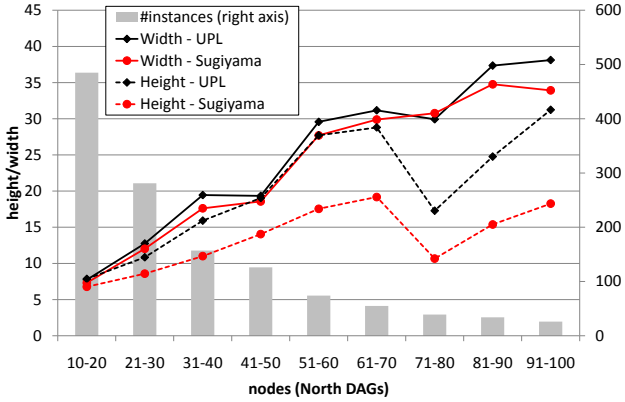


(c) random DAGs

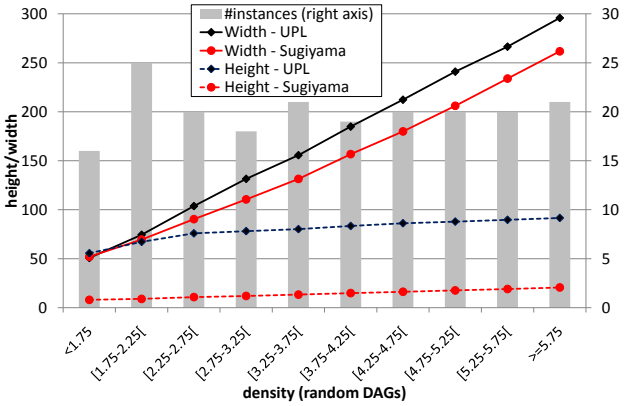
Figure 14: Comparison with Sugiyama's algorithm: number of crossings.



(a) Rome graphs

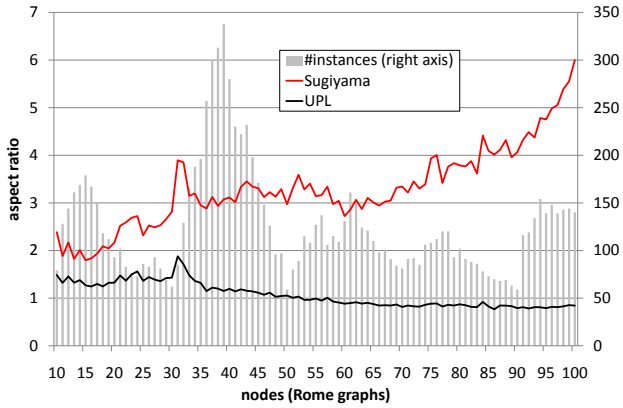


(b) North DAGs

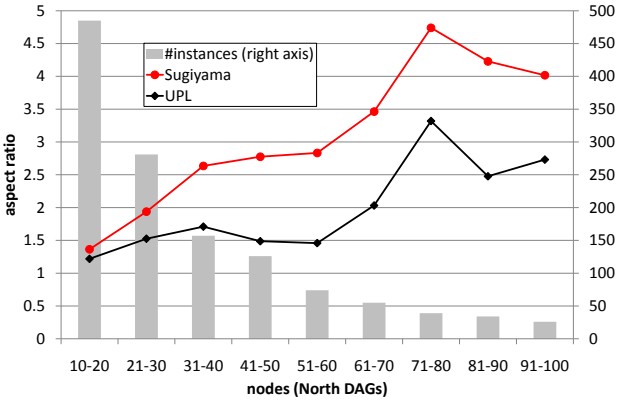


(c) random DAGs

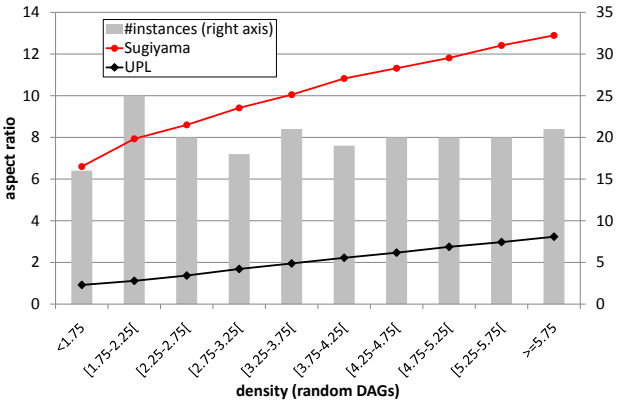
Figure 15: Comparison with Sugiyama's algorithm: height and width.



(a) Rome graphs

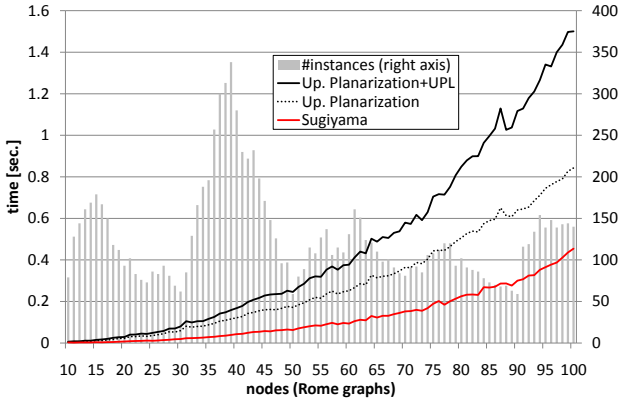


(b) North DAGs

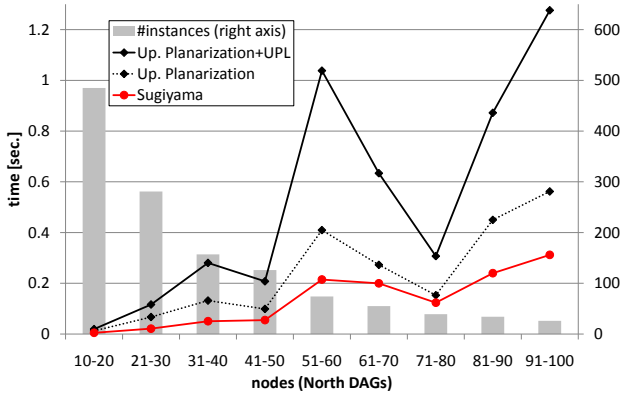


(c) random DAGs

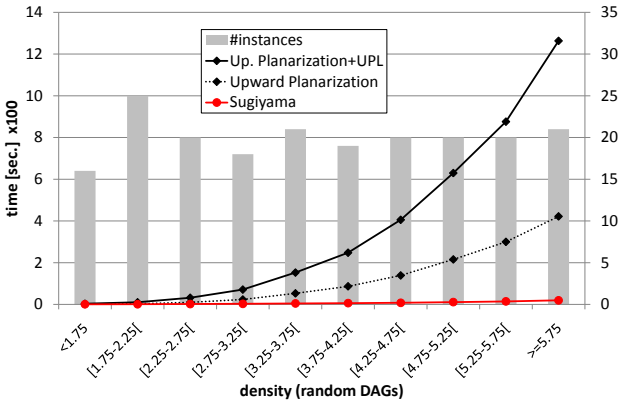
Figure 16: Comparison with Sugiyama's algorithm: aspect ratio.



(a) Rome graphs



(b) North DAGs



(c) random DAGs

Figure 17: Comparison with Sugiyama's algorithm: runtime.

with respect to this measure if a small number of crossings is an important issue.

UPL has certain advantages over Sugiyama's approach: A strong packing into few layers as produced by the Sugiyama approach will usually require a wider drawing than our planarization approach. Furthermore, such few layers can in fact be counterproductive regarding readability of the drawings; see, e.g., Fig. 1. Overall, we can observe that *UPL* obtains a more balanced aspect ratio than Sugiyama's approach; see Fig. 16.

In terms of running time (Fig. 17), we see that while Sugiyama's approach is generally faster, *UPL* is not too slow either, requiring below 1.5 seconds for the large instances of the Rome and North graphs. However, as the density of the graphs increases the gap between *Sugiyama* and *UPL* increases rapidly.

4 Conclusion

Traditional methods for drawing DAGs consider the number of crossings only as a second order priority. If it is of highest priority, one has to use algorithms based on upward planar representations. Our algorithm constitutes the first such algorithm that takes the special crossing nodes into account. As our experiments show, it generates drawings that are preferable over alternative methods for drawing upward planar representations. Furthermore, the new drawing algorithm is also comparable to Sugiyama's approach with respect to other quality measures, while offering a significantly smaller number of crossings.

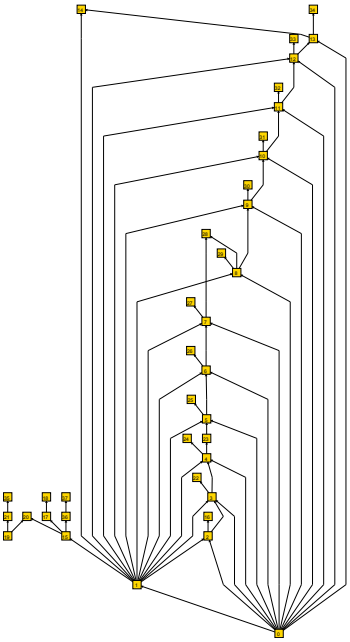
We conclude with two interesting open problems:

1. Regarding Lemma 1, can we archive a similar result for general (instead of only single-source) DAGs?

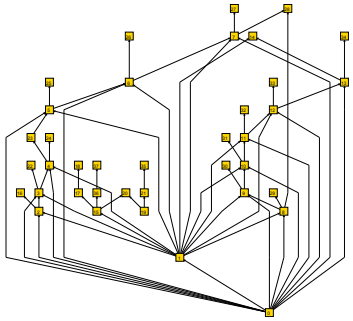
2. As shown by our experiments, there is a gap between Sugiyama's and our UPL approach regarding the height of the drawings. Clearly, this gap cannot be eliminated completely, but how can we reduce this gap? This problem includes the problem of finding a suitable upward planar representation R of the input DAG G for a compact realization.

5 Drawings

In this section we give some example drawings produced by our new algorithm *UPL* and the Sugiyama algorithm. For both drawing methods we use the LP-based layering and coordinate assignment algorithms proposed by Gansner et al. [13]. UPL uses the randomized upward crossing minimization (best of 20 runs) by Chimani et al. [8] and Sugiyama uses a barycenter heuristic for the crossing minimization step (best of 20 runs). All nodes in the drawings in Fig. 18 and Fig. 19 have the same sizes, while the nodes sizes in the drawings in Fig. 20 are set randomly.

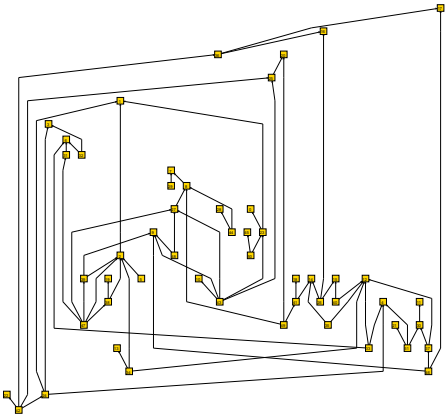


(a) UPL drawing of instance *g.38.21* (North DAGs) with one crossing.

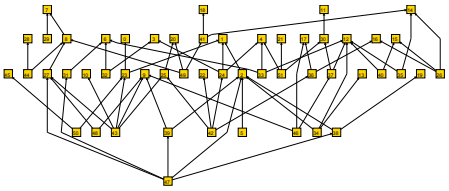


(b) Sugiyama drawing of instance *g.38.21* (North DAGs) with 27 crossings.

Figure 18: Comparison of drawings where all nodes have the same size.

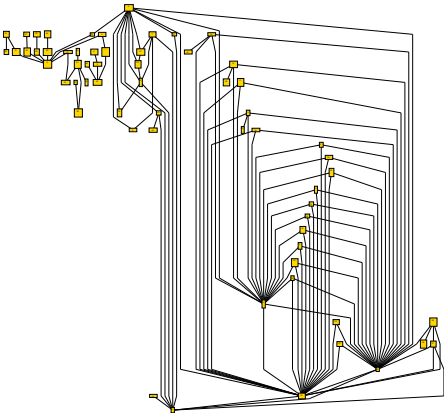


(a) UPL drawing of instance *grafo3579.53* (Rome graphs) with 9 crossings.

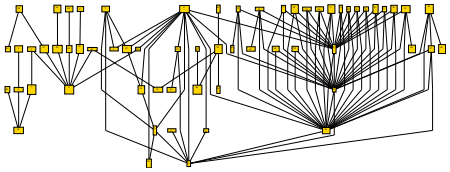


(b) Sugiyama drawing of instance *grafo3579.53* (Rome graphs) with 76 crossings.

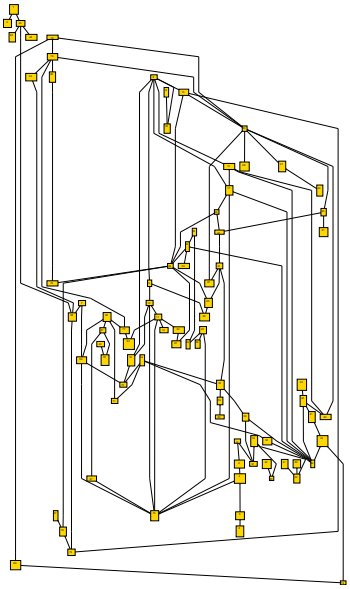
Figure 19: Comparison of drawings where all nodes have the same size.



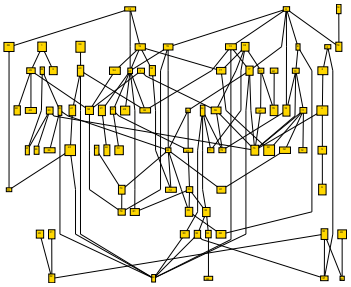
(a) UPL drawing of instance *g.60.0* (North DAGs) with 82 crossings.



(b) Sugiyama drawing of instance *g.60.0* (North DAGs) with 250 crossings.



(c) UPL drawing of instance *grafo8087.84* (Rome graphs) with 41 crossings.



(d) Sugiyama drawing of instance *grafo8087.84* (Rome graphs) with 134 crossings.

Figure 20: Comparison of drawings where the nodes have random size.

References

[1] C. Batini, M. Talamo, and R. Tamassia. Computer aided layout of entity relationship diagrams. *J. Syst. Software*, 4:163–173, 1984.

[2] G. D. Battista, E. Pietrosanti, R. Tamassia, and I. Tollis. Automatic layout of PERT diagrams with X-PERT. In *Proc. IEEE Workshop on Visual Languages*, pages 171–176, 1989.

[3] P. Bertolazzi, G. Di Battista, C. Mannino, and R. Tamassia. Optimal upward planarity testing of single-source digraphs. *SIAM J. Comput.*, 27(1):132–169, 1998.

[4] U. Brandes and B. Köpf. Fast and simple horizontal coordinate assignment. In *Proc. Graph Drawing '01*, pages 31–44, London, UK, 2002. Springer-Verlag.

[5] J. Branke, S. Leppert, M. Middendorf, and P. Eades. Width-restricted layering of acyclic digraphs with consideration of dummy nodes. *Inf. Process. Lett.*, 81(2):59–63, 2002.

[6] C. Buchheim, M. Jünger, and S. Leipert. A fast layout algorithm for k-level graphs. In *Proc. Graph Drawing '00*, volume 1984 of *LNCS*, pages 229–240. Springer, 1999.

[7] M. Chimani, C. Gutwenger, P. Mutzel, and H.-M. Wong. Layer-free upward crossing minimization. *ACM J. Exp. Algorithmics*. To appear.

[8] M. Chimani, C. Gutwenger, P. Mutzel, and H.-M. Wong. Layer-free upward crossing minimization. In *WEA 2008: Workshop on Experimental Algorithms*, volume 5038 of *LNCS*, pages 55–68. Springer, 2008.

[9] G. Di Battista, A. Garg, G. Liotta, A. Parise, R. Tamassia, E. Tassinari, F. Vargiu, and L. Vismara. Drawing directed acyclic graphs: An experimental study. *Int. J. Comput. Geom. Appl.*, 10(6):623–648, 2000.

[10] G. Di Battista, A. Garg, G. Liotta, R. Tamassia, E. Tassinari, and F. Vargiu. An experimental comparison of four graph drawing algorithms. *Comput. Geom. Theory Appl.*, 7(5-6):303–325, 1997.

[11] G. Di Battista, R. Tamassia, and I. G. Tollis. Area requirement and symmetry display of planar upward drawings. *Discrete Comput. Geom.*, 7(4):381–401, 1992.

[12] M. Eiglsperger, M. Kaufmann, and F. Eppinger. An approach for mixed upward planarization. *J. Graph Algorithms Appl.*, 7(2):203–220, 2003.

[13] E. Gansner, E. Koutsofios, S. North, and K.-P. Vo. A technique for drawing directed graphs. *Software Pract. Exper.*, 19(3):214–229, 1993.

[14] C. Gutwenger and P. Mutzel. An experimental study of crossing minimization heuristics. In *Proc. Graph Drawing '03*, volume 2912 of *LNCS*, pages 13–24, 2004.

[15] P. Healy and N. S. Nikolov. A branch-and-cut approach to the directed acyclic graph layering problem. In *Proc. Graph Drawing '02*, pages 98–109, London, UK, 2002. Springer-Verlag.

[16] P. Healy and N. S. Nikolov. How to layer a directed acyclic graph. In *Proc. Graph Drawing '01*, pages 16–30, London, UK, 2002. Springer-Verlag.

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

[18] OGDF – The Open Graph Drawing Framework. Technische Universität Dortmund, Chair of Algorithm Engineering; see `http://www.ogdf.net`.

[19] P. Rosenstiehl and R. E. Tarjan. Rectilinear planar layouts and bipolar orientations of planar graphs. *Discrete Comput. Geom.*, 1(1):343–353, 1986.

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