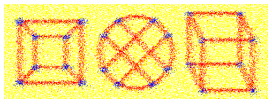

Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 10, no. 2, pp. 259–285 (2006)

Upward Planar Drawings and Switch-regularity Heuristics

Walter Didimo

Dipartimento di Ingegneria Elettronica e dell'Informazione
Università degli Studi di Perugia, Italy
http://www.diei.unipg.it/
didimo@diei.unipg.it

Abstract

In this paper we present a new characterization of *switch-regular* upward embeddings, a concept introduced by Di Battista and Liotta in 1998. This characterization allows us to define a new efficient algorithm for computing upward planar drawings of embedded planar digraphs. If compared with a popular approach described by Bertolazzi, Di Battista, Liotta, and Mannino, our algorithm computes drawings that are significantly better in terms of total edge length and aspect ratio, especially for low-density digraphs. Also, we experimentally prove that the running time of the drawing process is reduced in most cases.

Article Type	Communicated by	Submitted	Revised
regular paper	M. Kaufmann	March 2005	March 2006

This work is partially supported by the MIUR Project ALGO-NEXT: Algorithms for the Next Generation Internet and Web: Methodologies, Design and Applications

1 Introduction

The *upward drawing* convention is commonly adopted to display acyclic digraphs representing hierarchical structures, like for example PERT diagrams and class inheritance diagrams. In an upward drawing each vertex is represented as a point of the plane and all edges are drawn as curves monotonically increasing in a common direction (e.g., the vertical one). An *upward planar drawing* is an upward drawing with no edge crossing (see, e.g., Figure 1(b)). It is known that not all planar digraphs admit an upward planar drawing, and Garg and Tamassia [17] proved that the upward planarity testing problem is NP-complete in the general case. Many papers have been devoted to the design of efficient algorithms for computing upward planar drawings of specific families of planar digraphs, including *st*-digraphs [11, 12, 15], planar lattices [21, 22, 25], rooted trees [7, 8, 14, 23, 26, 27, 28], single-source digraphs [4, 19], outerplanar digraphs [24]. More references can be found in books and surveys on the subject (see, e.g., [9, 16, 20]).

A fundamental result about upward planar drawability is described by Bertolazzi et al. [3]; they proved that the upward planarity testing and drawability problem can be solved in polynomial time for every planar digraph, under the assumption that the planar embedding of the digraph is assigned as part of the input and can not be changed. More precisely, given a planar embedded digraph G, the authors introduce the concept of *upward planar embedding* of G, which is a labeled embedding that specifies the type of angles at source- and sink-vertices inside each face of G; the label of an angle informs if that angle will be "large" (greater than π) or "small" (less than π) in the final drawing. The authors prove that an upward planar drawing of G exists if and only if there exists an upward planar embedding of G with a specific set of properties; such an upward embedding can be computed in polynomial time if it exists. In the same paper, the authors describe an algorithm that computes a drawing of an upward planar embedded digraph G in two steps: `Step 1`: Construct an including planar *st*-digraph of G adding a suitable set of dummy edges; `Step 2`: Compute a polyline drawing of the *st*-digraph using standard techniques [9, 20], and then remove the dummy edges. It is important that the number of dummy edges added during `Step 1` is kept as small as possible. Indeed, dummy edges restrict the choices that can be performed in `Step 2` to determine a good layout of the digraph, and they also influence both the running time and space requirements of `Step 2`.

As far as we know, the upward planarity testing and drawing algorithm described in [3] is the only one existing in the literature for general embedded planar digraphs and no alternative augmentation technique has been proposed so far for solving `Step 1`.

More recently, several papers have been devoted to the design of exponential-time algorithms for solving the upward planarity testing and drawability problem in the variable embedding setting. Bertolazzi et al. [2] describe a branch-and-bound algorithm for biconnected digraphs, and experimentally prove that it is fast for many practical instances. Fixed Parameter Tractable algorithms for

general planar digraphs are described by Chan [6] and by Healy and Lynch [18].

In this paper we address the problem of computing upward planar drawings of general embedded planar digraphs using the strategy of Bertolazzi et al. [3], which works in two steps as recalled above. We focus on `Step 1`, and investigate a new approach for augmenting an upward planar embedded digraph to an including st-digraph. The main results of this paper are listed below:

- We provide a new characterization for the class of *switch-regular* upward embeddings. This class has been introduced by Di Battista and Liotta in [10]. Our characterization is related to the one given in [5] to define "turn-regular" orthogonal representations.
- We exploit the above characterization to design a new polynomial-time algorithm that augments a given upward planar embedded digraph to an including st-digraph.
- We experimentally prove that our augmentation algorithm significantly reduces the number of dummy edges added to determine the including st-digraph if compared with the technique in [3]. This reduction dramatically improves the total edge length and the aspect ratio of the computed drawings, especially for low-density digraphs, positively affecting the readability of the drawings. Also, a reduction of the running time taken by the whole drawing process is observed in most cases.

The remainder of the paper is structured as follows. In Section 2 we recall basic definitions and properties about upward planarity and switch-regularity. In Section 3 we present the new characterization for switch-regular upward embeddings. The new augmentation algorithm is described in Section 4. In Section 5 the results of an extensive experimental study are presented. Examples of drawings computed with the new algorithm are shown in Section 6. Conclusions and open problems are given in Section 7.

2 Preliminaries

We assume familiarity with basic concepts of graph drawing and graph planarity [9]. We concentrate on planar digraphs with a given planar embedding and use a notation that is slightly revised with respect to the one adopted in [3, 10].

2.1 Upward Planar Drawings

Let G be an embedded planar digraph. A drawing Γ of G is an *upward planar drawing* if: (i) it has no edge crossing; (ii) it preserves the embedding of G; (iii) all the edges of G are drawn as curves monotonically increasing in the vertical direction. If G admits an upward planar drawing, it is called an *upward planar digraph*.

A vertex of G is *bimodal* if the circular list of its incident edges can be partitioned into two (possibly empty) lists, one consisting of incoming edges and the other consisting of outgoing edges. If all vertices of G are bimodal then G and its embedding are called *bimodal*. Acyclicity and bimodality are necessary conditions for the upward planar drawability of an embedded planar digraph [3]. However, they are not sufficient conditions in general.

Let f be a face of an embedded planar bimodal digraph G and suppose that the boundary of f is visited counterclockwise. Let $s = (e_1, v, e_2)$ be a triplet such that v is a vertex of the boundary of f and e_1, e_2 are incident edges of v that are consecutive on the boundary of f. Triplet s is called a *switch of* f if the direction of e_1 is opposite to the direction of e_2 (note that e_1 and e_2 may coincide if G is not biconnected). If e_1 and e_2 are both incoming in v, then s is a *sink-switch of* f; if they are both outgoing, s is a *source-switch of* f. Observe that the number of source-switches of f is equal to the number of sink-switches of f. Let $2n_f$ be the total number of switches of f (both source- and sink-switches); the *capacity of* f is defined as $c_f = n_f - 1$ if f is an internal face, and $c_f = n_f + 1$ if f is the external face.

An assignment of the sources and sinks of G to its faces is *upward consistent* if the following properties hold: (a) a source (sink) is assigned to exactly one of its incident faces; (b) for each face f, the number of sources and sinks assigned to f is equal to c_f.

The following theorem gives a characterization of the class of embedded digraphs that are upward planar.

Theorem 1 [3] *Let G be an embedded planar bimodal digraph. G is upward planar if and only if it admits an upward consistent assignment.*

If G has an upward consistent assignment then the *upward planar embedding of* G corresponding to that assignment is a labeled planar embedding of G such that for each face f and for each switch $s = (e_1, v, e_2)$ of f we have that: (i) s is labeled L if v is a source or a sink assigned to f; (ii) s is labeled S otherwise.

If f is a face of an upward planar embedding, the circular list of labels of f is denoted by σ_f. Also, S_{σ_f} and L_{σ_f} denote the number of S and L labels of f, respectively.

Property 1 [3] *If f is a face of an upward planar embedding then $S_{\sigma_f} = L_{\sigma_f} + 2$ if f is internal, and $S_{\sigma_f} = L_{\sigma_f} - 2$ if f is external.*

Given an upward planar embedding of a digraph G, it is possible to construct an upward planar drawing of G such that every angle at a source-switch or a sink-switch of f is greater than π when the switch is labeled L, and less than π when the switch is labeled S. Figures 1(a) and 1(b) show an upward planar embedded digraph and a corresponding upward planar drawing, respectively.

As recalled in the introduction, the computation of an upward planar drawing of G, starting from an upward planar embedding of G, is done by first constructing a planar st-digraph that includes G [3]; this step is usually called

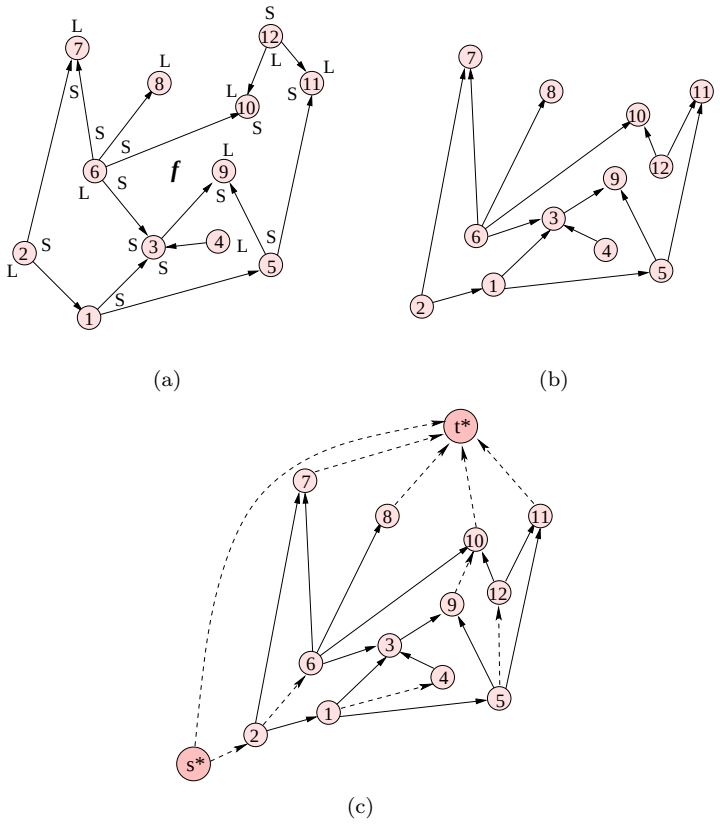


Figure 1: (a) A bimodal digraph G with a given upward planar embedding. (b) An upward planar drawing of G corresponding to the upward embedding. (c) An st-digraph including G; the dashed edges that are not incident on s^* and t^* form a complete saturator of G.

saturating phase. After that, a drawing of the st-digraph is computed with standard techniques and dummy vertices and edges added in the saturating phase are eventually removed.

To determine an st-digraph that includes G, G is augmented with a new source vertex s^*, a new sink vertex t^*, the edge (s^*, t^*), and a suitable set of dummy edges, called *saturating edges*. More formally, the saturating edges can be added applying the following rules.

- If $s = (e_1, v, e_2)$ and $s' = (e'_1, v', e'_2)$ are two source-switches of a face f such that s is labeled S and s' is labeled L (see Figure 2(a)), then we can add a saturating edge $e = (v, v')$ splitting f into two faces f' and f''. Face f' contains the new source-switch (e_1, v, e) labeled S, and f'' contains the new source-switch (e, v, e_2) labeled S. Also, v' does not belong to any

switch labeled L in f' and f''. We say that s *saturates* s' and the new embedding is still upward planar.

- If $s = (e_1, v, e_2)$ and $s' = (e'_1, v', e'_2)$ are two sink-switches of a face f such that s is labeled L and s' is labeled S (see Figure 2(b)), then we can add a saturating edge $e = (v, v')$ splitting f into two faces f' and f''. Face f' contains the new sink-switch (e, v', e'_2) labeled S, and f'' contains the new sink-switch (e'_1, v', e) with label S. Also, v does not belong to any switch labeled L in f' and f''. We say that s' *saturates* s and the new embedding is still upward planar.

- Once all faces have been decomposed by using the two rules above, we can add dummy edges that either connect a sink-switch labeled L of the external face to t^*, or s^* to a source-switch labeled L of the external face. After the insertion of these edges the embedding is still upward planar.

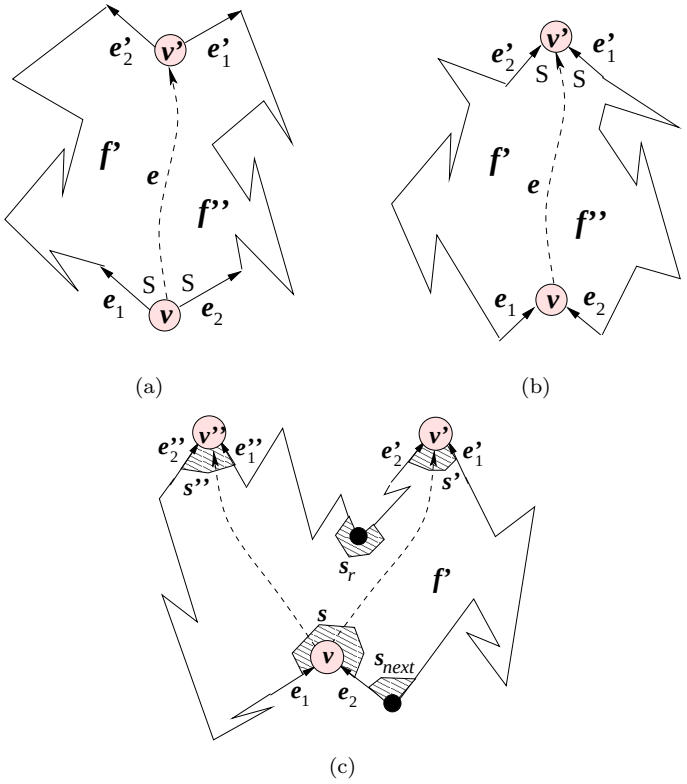


Figure 2: (a) A saturating edge between two source-switches. (b) A saturating edge between two sink-switches. (c) Illustration of the proof of Theorem 3.

A *saturator* of a face f of G is a set of saturating edges that split f. Such a saturator is *complete* if no more saturating edges can be added to decompose f. A set of saturating edges that is complete for all faces of G is called a *complete saturator* of G. Notice that the set of saturating edges added to determine an st-digraph including G is a complete saturator of G. Figure 1(c) shows an st-digraph including the upward planar embedded digraph in Figure 1(a); the dashed edges that are not incident on s^* and t^* form a complete saturator.

In Section 4 we recall a classical algorithm that computes a complete saturator of an upward planar embedded digraph G, and then we propose a new algorithm.

2.2 Switch-regularity

An internal face f of an upward planar embedding is *switch-regular* if σ_f does not contain two distinct maximal subsequences σ_1 and σ_2 of S labels such that $S_{\sigma_1} > 1$ and $S_{\sigma_2} > 1$. The external face f is *switch-regular* if σ_f does not contain two consecutive S labels. An upward planar embedding is *switch-regular* if all its faces are switch-regular. For example, the upward planar embedding of Figure 1(a) is not switch-regular, since face f is not switch-regular. All the other faces are switch-regular.

The following result establishes an interesting connection between switch-regular faces and complete saturators of an upward planar embedded digraph.

Theorem 2 [10] *Let f be a face of an upward planar embedding of a digraph G. Face f has a unique complete saturator if and only if f is switch-regular.*

Theorem 2 implies that an upward planar embedded digraph has a unique complete saturator if and only if it is switch-regular. In this case there is only one way to augment G to become an including st-digraph.

3 A New Characterization of Switch-regular Upward Embeddings

In this section we give a new characterization of switch-regular upward embeddings; it is strongly related to the definition of turn-regular orthogonal representations given in [5].

Let G be an embedded bimodal planar digraph with a given upward planar embedding. Let f be a face of G. A *reflex switch* of f is a switch of f with label L. A *convex switch* of f is a switch of f with label S. Denote by Σ_f the circular list of switches of f while visiting the boundary of f counterclockwise (clearly, $|\Sigma_f| = |\sigma_f|$). For any switch $s \in \Sigma_f$, we define $turn(s) = -1$ if s is reflex, and $turn(s) = 1$ if s is convex.

Let $s' = (e'_1, v, e'_2)$ and $s'' = (e''_1, v'', e''_2)$ be two switches in Σ_f. Denote by $\Sigma_f(s', s'')$ the subsequence of Σ_f from s' (included) to s'' (excluded). We define the following function:

$$rotation_f(s', s'') = \sum_{s \in \Sigma_f(s', s'')} turn(s).$$

Notice that, by Property 1, $rotation_f(s, s) = +2$ for any switch s of an internal face f. If f is external, then $rotation_f(s, s) = -2$. Also $rotation_f(s', s'') = rotation_f(s', s) + rotation_f(s, s'')$, for each ordered sequence $s', s, s'' \in \Sigma_f$. Let $\{s', s''\}$ be an ordered pair of reflex switches of f. We call $\{s', s''\}$ a pair of *kitty corners of* f if one of the following holds:

- $rotation_f(s', s'') = +1$.
- $rotation_f(s', s'') = -3$ and f is external.

By Property 1, if $\{s', s''\}$ is a pair of kitty corners of a face f (internal or external), then $\{s'', s'\}$ is a pair of kitty corners of f, too. Indeed, if f is internal and $rotation_f(s', s'') = +1$, then $rotation_f(s'', s') = +1$. Also if f is external and $rotation_f(s', s'') = +1$, then $rotation_f(s'', s') = -3$. In the upward planar embedding of Figure 1(a), for $s' = ((3, 9), 9, (5, 9))$ and $s'' = ((12, 11), 12, (12, 10))$ in face f, we have $rotation_f(s', s'') = +1$, and therefore s' and s'' are kitty corners of f. The following theorem is the main result of this section.

Theorem 3 *A face of an upward planar embedding is switch-regular if and only if it has no kitty corners.*

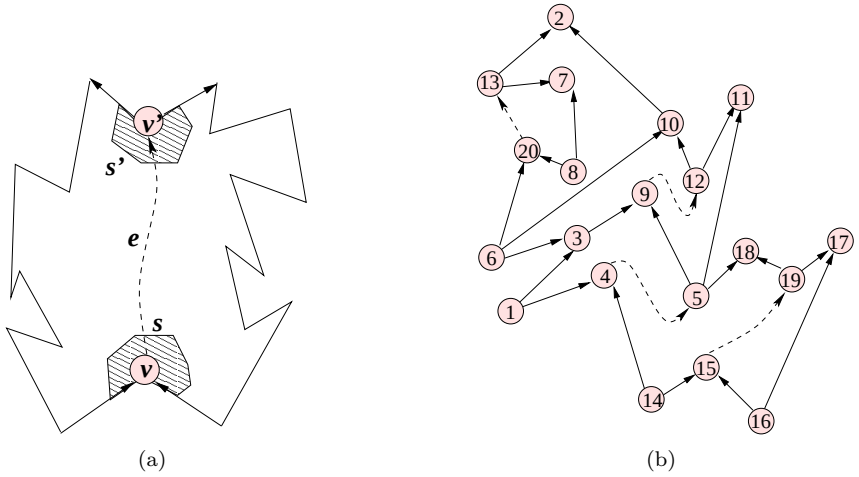
Proof: We prove the statement for an internal face f. A similar proof applies for the external face. Let f be a switch-regular face. Suppose by contradiction that f contains a pair $\{s', s''\}$ of kitty corners and consider the subsequence $\Sigma_f(s', s'')$. Since $rotation_f(s', s'') = +1$, then in $\Sigma_f(s', s'')$ the number of convex switches is equal to the number of reflex switches plus one. Therefore, since s' is a reflex switch, in $\Sigma_f(s', s'')$ there are necessarily two consecutive switches labeled S. Applying the same reasoning, there must be two consecutive switches with label S in the subsequence $\Sigma_f(s'', s')$. Since s' and s'' are labeled L, we have found two maximal subsequences of S labels both of size greater than one. Therefore, f is not switch-regular, a contradiction.

Conversely, let f be a face that does not contain kitty corners. Suppose by contradiction that f is not switch-regular. By Theorem 2 f has no unique saturator. This implies that in f there is a reflex switch $s = (e_1, v, e_2)$ that can be saturated by at least two distinct convex switches, say $s' = (e'_1, v', e'_2)$ and $s'' = (e''_1, v'', e''_2)$. Assume, without loss of generality, that s, s', and s'' are sink-switches. Each of the two saturating edges (v, v') and (v, v'') would split f, keeping the embedding upward planar. Refer to the notation of Figure 2(c): Denote by s_{next} the switch that follows s in Σ_f, and let f' be the face to the right of the saturating edge (v, v'). Since the complete rotation of an internal face is always 2, we have that $rotation_{f'}(s_{next}, s') = 2 - rotation_{f'}(s', s_{next}) = +1$. Also, since s is a reflex switch of f, we have that $rotation_f(s, s') = 0$. By a similar reasoning applied on the face to the right of the saturating edge (v, v''),

we have that $rotation_f(s, s'') = 0$, and hence $rotation_f(s', s'') = 0$. This implies that in the subsequence $\Sigma_f(s', s'')$ there exists at least one reflex switch s_r such that $rotation_f(s, s_r) = +1$ (indeed s' is labeled S, and the number of L labels in $\Sigma_f(s', s'')$ is equal to the number of S labels). Therefore, $\{s, s_r\}$ is a pair of kitty corners, a contradiction. □

4 A Switch-regularity Heuristic

Let G be a planar bimodal digraph with a given upward planar embedding. In [3] the authors describe a linear-time algorithm that constructs an st-digraph including G by computing a complete saturator of G. This algorithm recursively decomposes every face f, searching on the boundary of f subsequences of three consecutive switches s, s', s'' such that both s and s' are labeled S, while s'' is labeled L. Each time such a sequence is found, the algorithm splits f into two faces by adding a saturating edge connecting the vertices of s and s''. When there are no more subsequences of labels SSL in a face, then the algorithm connects the dummy source s^* to each reflex source-switch of the external face, and each reflex sink-switch of the external face to the dummy sink t^*. In the remainder of the paper we call this algorithm `SimpleSat`.



(a) (b)

Figure 3: (a) A face having kitty corners s, s' is split by edge $e = (v, v')$. (b) An upward planar embedding augmented to a switch-regular one; dummy edges are dashed.

We use the characterization of Theorem 3 to design a new heuristic for computing a planar st-digraph including G. For each face f, we test if f is switch-regular or not. If f is not switch-regular, we apply an $O(deg(f))$ procedure that finds a pair $\{s, s'\}$ of kitty corners of f, where $deg(f)$ is the degree of f (i.e. the number of its edges). Such a procedure uses the same technique described in [5]

for detecting a pair of kitty corners in a face of an orthogonal representation. Once $\{s, s'\}$ is detected, we split f by adding to G a dummy edge connecting the vertices of s and s' (see Figure 3(a)). After the insertion of this edge, the new embedding is still an upward planar embedding. We recursively decompose all faces that are non-switch-regular by applying the above algorithm to make the upward planar embedding switch-regular. Figure 3(b) shows an upward planar embedding that is augmented to a switch-regular one by using the above strategy. When the upward planar embedding becomes switch-regular, we apply the `SimpleSat` algorithm to add the edges that are still needed to construct an st-digraph including G. Notice that, since we apply algorithm `SimpleSat` on a switch-regular upward embedding, the complete saturator determined by this algorithm is uniquely defined. We call our heuristic `SrSat`.

Regarding time complexity, `SrSat` takes $O(n^2)$ time in the worst case, since there may be an $O(n)$ number of kitty corners, and the detection of each pair requires $O(n)$ time. In practice however, since `SrSat` adds less edges than `SimpleSat`, the overall running time of the drawing algorithm is reduced (see Section 5).

5 Experimental results

We implemented and tested heuristics `SimpleSat` and `SrSat` on a large test suite of graphs, in order to compare their performances. The test suite consists of two subsets, `Small-graphs` and `Large-graphs`, of upward planar embedded digraphs. Subset `Small-graphs` contains 800 digraphs having number of vertices in $\{10, 20, \ldots, 100\}$ and density in $\{1.2, 1.4, 1.6, 1.8\}$ (the density is the ratio between the number of edges and the number of vertices). Subset `Large-graphs` contains 160 digraphs, each having number of vertices in the set $\{500, 600, \ldots, 1500, 1600, \ldots, 2000\}$ and density ranging from 1.2 to 1.3. All digraphs have been randomly generated. However, since these digraphs are connected, upward planar, and they have specific given density values, the design of the generation algorithm was a difficult task. We used two different approaches to generate the digraphs for the two subsets; both the approaches guarantee that the digraphs have the required properties. Each graph in subset `Small-graphs` was generated by the following procedure: (i) Generate, with a uniform probability distribution, a connected (possibly non-planar) graph having the desired number of vertices; (ii) Compute a spanning tree of this graph by randomly choosing the root vertex; (iii) Randomly add edges to the spanning tree until the desired density value is reached: each time a new edge is chosen for insertion, this edge is really added only if the planarity of the graph is not violated, otherwise the edge is discarded; (iv) Once a planar embedded graph has been generated, an upward orientation is assigned to the graph by applying the network-flow algorithm described in [13].

The above generation algorithm requires the application of a planarity testing each time a new edge is chosen for possible insertion. Therefore, it does not allow the generation of large graphs in a reasonable time. In order to generate

graphs for the subset `Large-graphs` we used a different and faster algorithm. First we generate a planar embedded biconnected graph within a certain range of density values by applying the technique described in [1], and then we assign an orientation to the graph by using the algorithm in [13]. Hence, the graphs in this subset are always biconnected.

All experiments were performed on a PC Pentium M, 1.6 Ghz, 512 MB RAM, and Linux OS.

5.1 Structural measures

From the graph structure point of view, we measured:

- The percentage of non-switch-regular faces in the upward planar embedded digraphs of our test suite;
- The percentage of kitty corners detected and saturated by `SrSat` over to the total number of reflex vertices (i.e., number of sources and sinks);
- The total number of dummy edges added by `SimpleSat` and `SrSat` to compute an including *st*-digraph.

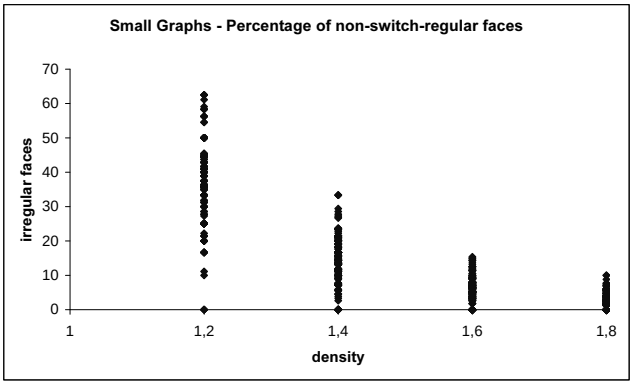
The first two parameters provide an indication of how much the upward planar embedded digraphs are non-switch-regular. As already found in other similar experiments on orthogonal representations [5], the percentage of non-switch-regular faces decreases exponentially with the increasing of the graph density (see Figure 4). For small graphs of density 1.2 we have up to 60% of faces that are not switch-regular (Figure 4(a)). The percentage of non-switch-regular faces in the large graphs of our test suite is 33% in the average (Figure 4(b)). This behavior is confirmed by the percentage of kitty corners, which decreases with the increasing of the graph density (Figure 5). In particular, for small graphs (Figure 5(a)), the average percentages are 34%, 21%, 12%, and 8%, for densities $1.2, 1.4, 1.6$, and 1.8, respectively.

Figure 6 shows the number of dummy edges added by the two heuristics for low density small graphs and for large graphs. Heuristic `SrSat` adds about 16% of edges less than `SimpleSat` in the average. This improvement is significantly attenuated for graphs with high density, since the number of non-switch-regular faces and the number of kitty corners in these graphs is quite small; for graphs with density 1.8, we have a 4% reduction of dummy edges in the average (the chart is omitted).

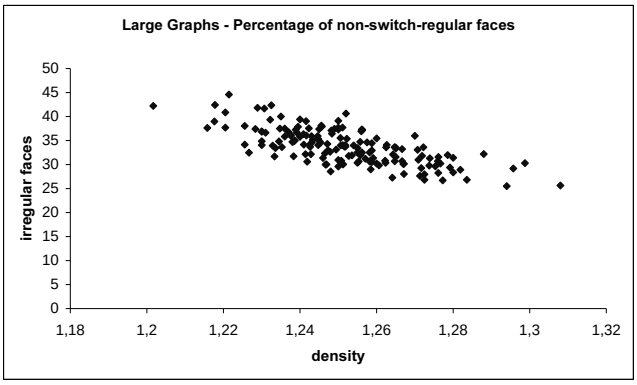
Since the performances of algorithms `SimpleSat` and `SrSat` differ especially on low density graphs (due to the high number of non-switch-regular faces), in the following we only discuss the results for low density small graphs and for large graphs.

5.2 Aesthetics

Concerning drawing aesthetics we measured the total edge length, the area, and the aspect ratio of the computed drawings.



(a)

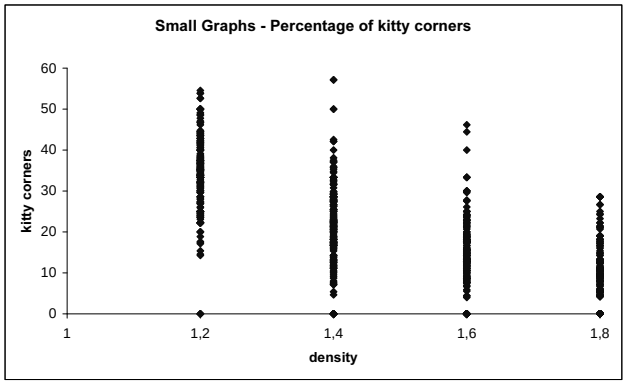


(b)

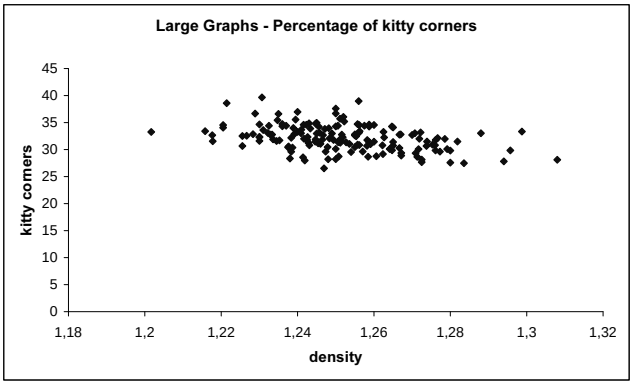
Figure 4: Percentage of non-switch-regular faces (y-axis) with respect to the graph density (x-axis): (a) `Small-graphs`; (b) `Large-graphs`.

While the areas of the drawings computed by `SrSat` and `SimpleSat` are quite similar (Figure 7), `SrSat` dramatically improves both the total edge length and the aspect ratio with respect to `SimpleSat`. This improvement has a strong impact on the readability of the drawings, as shown by some examples in Section 6.

More in detail, the average reduction for total edge length is about 12% for small graphs (Figure 8(a)) and it grows to 27% for large graphs (Figure 8(b)). Concerning the aspect ratio, we measured the deviation of the width-to-height ratio of the drawings computed by the two heuristics from an "optimum" aspect ratio of 4/3, i.e., the aspect ratio of a common workstation screen (Figure 9). The deviation from the optimum aspect ratio is rather small for the drawings computed by `SrSat`, while it is usually remarkable for the drawings computed by `SimpleSat`.



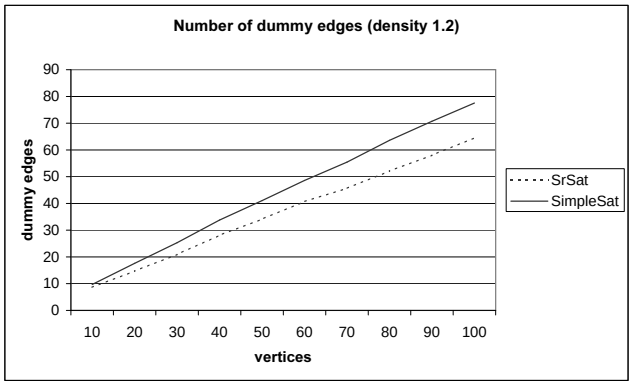
(a)



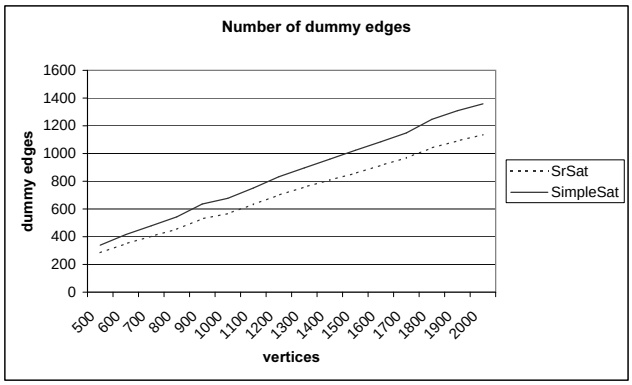
(b)

Figure 5: Percentage of kitty corners saturated by `SrSat` (y-axis) over the total number of reflex vertices (x-axis): (a) `Small-graphs`; (b) `Large-graphs`.

The intuition behind the better performance of `SrSat` in terms of aspect ratio and total edge length is shown in Figure 10. The figure depicts two upward drawings of the same non-switch-regular face, obtained by decomposing the face into smaller st-faces using heuristic `SimpleSat` (Figure 10(a)) and heuristic `SrSat` (Figure 10(b)); the dummy edges are dashed and have a light color. `SrSat` inserts dummy edges that connect kitty corners and that force them to have a "bottom-top" relative position in the final upward drawing; this determines a better balancing between the width and the height of the drawing. Conversely, heuristic `SimpleSat` inserts dummy edges between S labeled switches and L labeled switches; this typically stretches the drawing in the horizontal direction, causing long edges and poor aspect ratio.



(a)



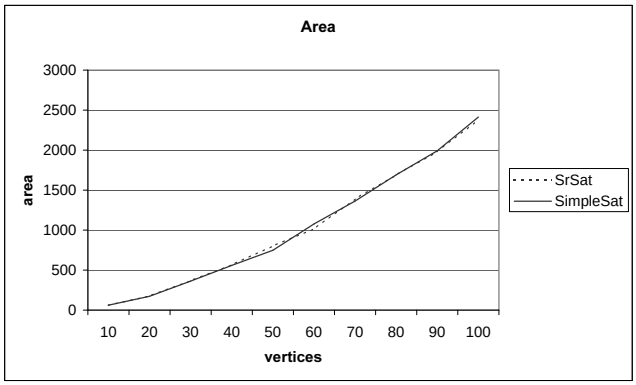
(b)

Figure 6: Average number of dummy edges added by the two heuristics with respect to the number of vertices: (a) `Small graphs`; (b) `Large graphs`.

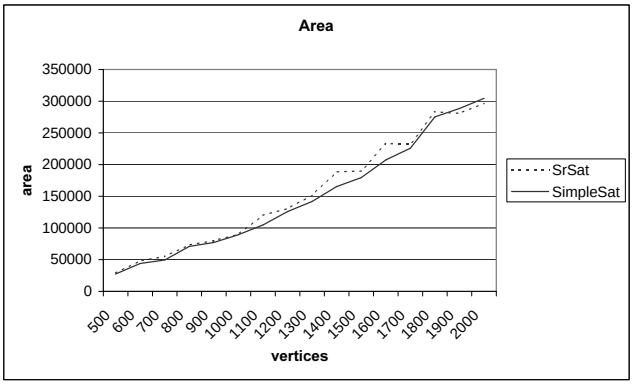
5.3 Efficiency

Although the asymptotic cost of `SrSat` is $O(n^2)$, in practice this heuristic causes a reduction of the running time taken by the whole drawing process, due to the "small" number of dummy edges added with respect to `SimpleSat`.

In fact, on the *st*-digraphs computed by the two heuristics we applied the same compaction algorithm to determine a polyline drawing of the digraph (see e.g. [9]). This algorithm first computes a visibility representation of the *st*-digraph, then applies on it an $O(n^2 \log n)$-time min-cost-flow technique to minimize the total edge length, and finally constructs a polyline drawing from the compact visibility representation. We measured both the CPU time required by the two saturating heuristics and the overall CPU time spent for computing the upward planar drawings using the compaction algorithm described above. We observed that `SimpleSat` is about 48% faster than `SrSat` on large graphs, which



(a)

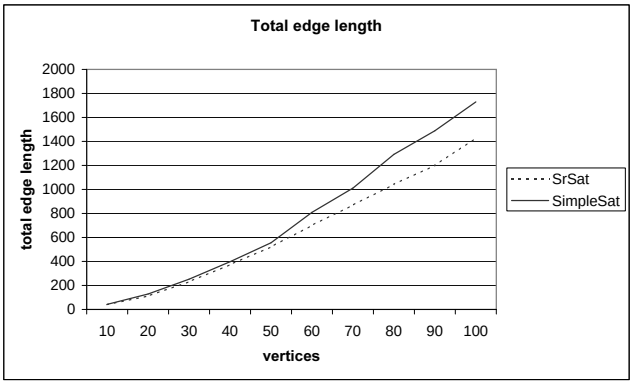


(b)

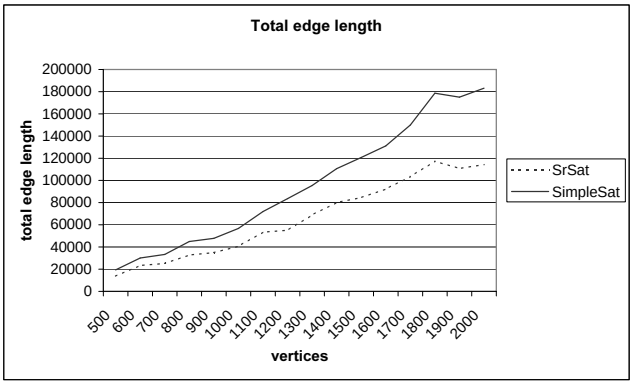
Figure 7: Average areas of the drawings computed by the two heuristics with respect to the number of vertices: (a) `Small graphs`; (b) `Large graphs`.

confirms the theoretical asymptotic complexity of the two heuristics. Nevertheless, both the saturating heuristics are very fast, and in most cases they take less than 0,5% of the CPU time required by the whole drawing process. This implies that, in practice, the extra time required by `SrSat` is negligible if compared with the benefits of having less dummy edges in the rest of the process.

For the considerations above, we only report the charts for the overall CPU time. While for small graphs the choice of the saturating heuristic has no remarkable effect on the CPU time, which is always significantly less than 1 second (Figure 11(a)), applying `SrSat` against `SimpleSat` on large graphs reduces the overall CPU time by about 10% in the average (Figure 11(b)).



(a)

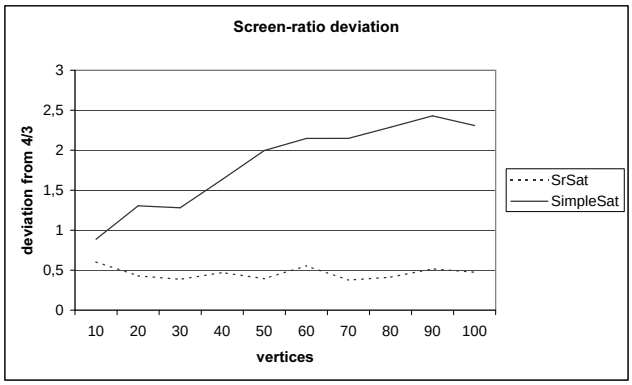


(b)

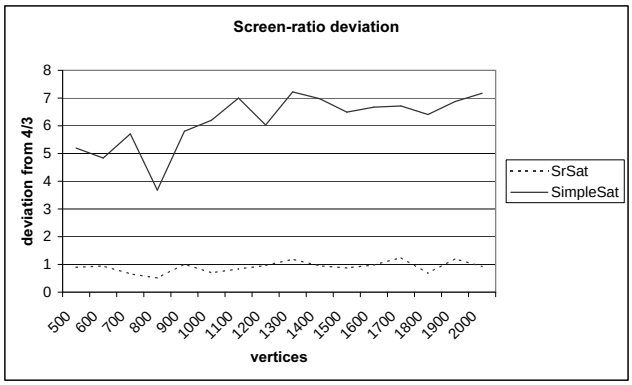
Figure 8: Average total edge length of the drawings computed by the two heuristics with respect to the number of vertices: (a) `Small graphs`; (b) `Large graphs`.

6 Drawing Gallery

In this section we report a small drawing gallery that shows how the better aesthetics of the drawings computed using `SrSat` positively affect the readability. All drawings are scaled down to perfectly fit in a rectangle with 4/3 aspect ratio. We remark how the good aspect ratio of the drawings computed by `SrSat` allows us to visualize these drawings with a size that is bigger than the one for the drawings computed by `SimpleSat`. This improvement can be better appreciated for large graphs (see, e.g., Figures 16 and 17).



(a)



(b)

Figure 9: Deviation from the 4/3 aspect ratio for the drawings computed by the two heuristics: (a) `Small graphs`; (b) `Large graphs`.

7 Conclusions and Open Problems

In this paper we presented a new algorithm for computing upward planar drawings of digraphs. This algorithm exploits a novel saturating strategy for augmenting an upward planar embedding to an including *st*-digraph, and it is based on a new characterization of switch-regular faces.

We experimentally proved that, if compared with a popular technique known in the literature, our saturating heuristic significantly reduces the number of dummy edges added to determine an including *st*-digraph. This reduction positively affects the drawing process, both in terms of drawing readability and in terms of running time and space. In particular, for low-density digraphs, the new algorithm dramatically outperforms the classical technique in terms of total edge length and aspect ratio of the computed drawings. Beyond the positive results of the new technique, several problems are still open. Some of them are

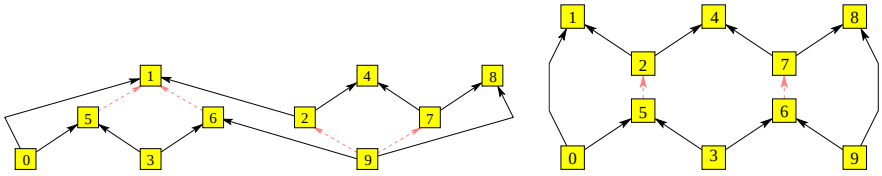
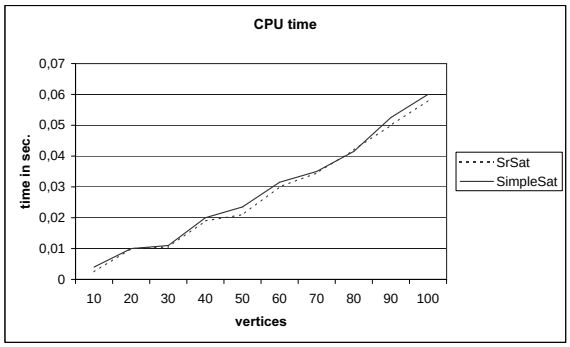
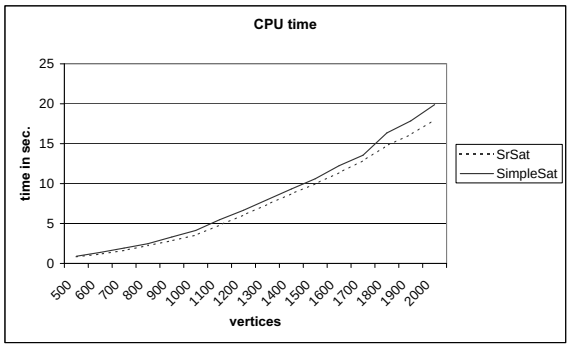


Figure 10: A figure that illustrates why the use of `SrSat` gives rise to upward drawings having better aspect ratio and total edge length than those computed using `SimpleSat`. The figure shows two different upward drawings of the a non-switch-regular face; in (a) the face has been decomposed using `SimpleSat`, while in (b) the face has been decomposed using `SrSat`. The dummy edges inserted by the two heuristics are dashed and have a light color.



(a)



(b)

Figure 11: CPU time (in seconds) required for computing upward planar drawings applying the two different heuristics in the saturating phase: (a) `Small graphs`; (b) `Large graphs`.

listed below:

- The implementation of our saturating heuristic currently requires $O(n^2)$ time, where n is the number of vertices of the digraph. Is it possible to design a linear time implementation?
- Our heuristic recursively detects and saturates pairs of kitty corners in a face. When multiple pairs occur in the same face, saturating a pair in place of another may influence the remainder of the saturating phase. Is it possible to design more effective heuristics that are capable to detect at each step the "best" pair of kitty corners to be saturated?
- What about drawing algorithms that also improve the area of the drawing?

Acknowledgements

We thank the anonymous referees for their useful comments.

References

[1] P. Bertolazzi, G. Di Battista, and W. Didimo. Computing orthogonal drawings with the minimum number of bends. *IEEE Transactions on Computers*, 49(8):826–840, 2000.

[2] P. Bertolazzi, G. Di Battista, and W. Didimo. Quasi-upward planarity. *Algorithmica*, 32(3):474–506, 2002.

[3] P. Bertolazzi, G. Di Battista, G. Liotta, and C. Mannino. Upward drawings of triconnected digraphs. *Algorithmica*, 6(12):476–497, 1994.

[4] P. Bertolazzi, G. Di Battista, C. Mannino, and R. Tamassia. Optimal upward planarity testing of single-source digraphs. *SIAM Journal on Computing*, 27:132–169, 1998.

[5] S. Bridgeman, G. Di Battista, W. Didimo, G. Liotta, R. Tamassia, and L. Vismara. Turn-regularity and optimal area drawings of orthogonal representations. *Computational Geometry: Theory and Applications*, 16:53–93, 2000.

[6] H. Chan. A parameterized algorithm for upward planarity testing. In *Algorithms (Proc. ESA '04)*, volume 3221 of *Lecture Notes Computer Science*, pages 157–168, 2004.

[7] P. Crescenzi, G. Di Battista, and A. Piperno. A note on optimal area algorithms for upward drawings of binary trees. *Computational Geometry: Theory and Applications*, 2:187–200, 1992.

[8] P. Crescenzi, P. Penna, and A. Piperno. Linear area upward drawings of AVL trees. *Computational Geometry: Theory and Applications*, 9(1-2):25–42, 1998.

[9] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing*. Prentice Hall, Upper Saddle River, NJ, 1999.

[10] G. Di Battista and G. Liotta. Upward planarity checking: "faces are more than polygons". In *Graph Drawing (Proc. GD '98)*, volume 1547 of *Lecture Notes Computer Science*, pages 72–86, 1998.

[11] G. Di Battista and R. Tamassia. Algorithms for plane representations of acyclic digraphs. *Theoretical Computer Science*, 61:175–198, 1988.

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

[13] W. Didimo and M. Pizzonia. Upward embeddings and orientations of undirected planar graphs. *Journal of Graph Algorithms and Applications*, 7(2):221–241, 2003.

[14] A. Garg, M. T. Goodrich, and R. Tamassia. Planar upward tree drawings with optimal area. *International Journal on Computational Geometry and Applications*, 6:333–356, 1996.

[15] A. Garg and R. Tamassia. Efficient computation of planar straight-line upward drawings. In *Graph Drawing '93 (Proc. ALCOM Workshop on Graph Drawing)*, 1993.

[16] A. Garg and R. Tamassia. Upward planarity testing. *Order*, 12:109–133, 1995.

[17] A. Garg and R. Tamassia. On the computational complexity of upward and rectilinear planarity testing. *SIAM Journal on Computing*, 31(2):601–625, 2001.

[18] P. Healy and K. Lynch. Fixed-parameter tractable algorithms for testing upward planarity. In *Theory and Practice of Computer Science (Proc. SOFSEM '05)*, volume 3381 of *Lecture Notes Computer Science*, pages 199–208, 2005.

[19] M. D. Hutton and A. Lubiw. Upward planarity testing of single-source acyclic digraphs. *SIAM Journal on Computing*, 25(2):291–311, 1996.

[20] M. Kaufmann and D. Wagner. *Drawing Graphs*. Springer Verlag, 2001.

[21] D. Kelly. Fundamentals of planar ordered sets. *Discrete Mathematics*, 63:197–216, 1987.

[22] D. Kelly and I. Rival. Planar lattices. *Canadian Journal of Mathematics*, 27(3):636–665, 1975.

[23] S. K. Kim. Simple algorithms for orthogonal upward drawings of binary and ternary trees. In *Proc. 7th Canadian Conference on Computational Geometry*, pages 115–120, 1995.

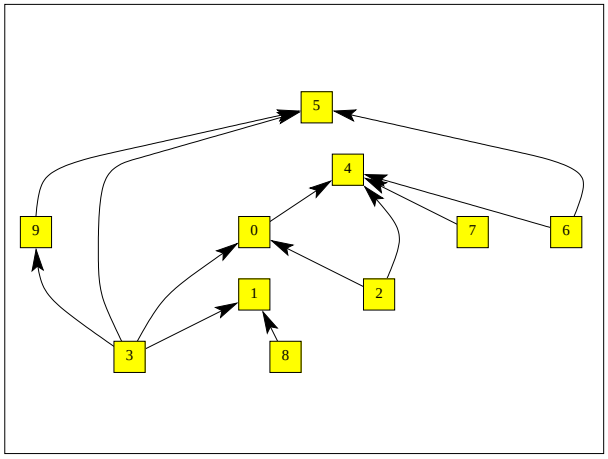
[24] A. Papakostas. Upward planarity testing of outerplanar dags. In *Graph Drawing (Proc. GD '95)*, volume 894 of *Lecture Notes Computer Science*, pages 7298–306, 1995.

[25] C. Platt. Planar lattices and planar graphs. *Journal of Combinatorial Theory Series B*, 21:30–39, 1976.

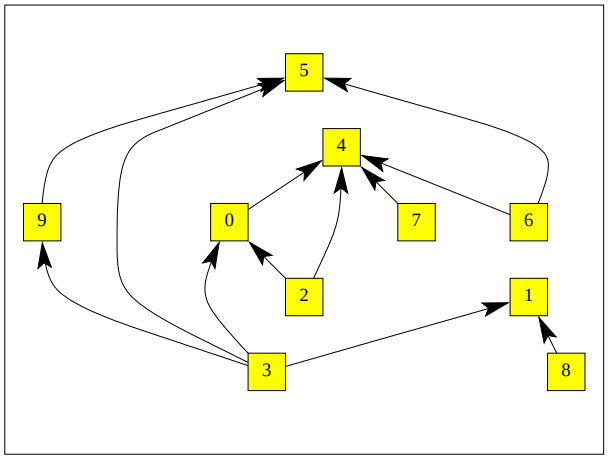
[26] E. Reingold and J. Tilford. Tidier drawing of trees. *IEEE Transactions on Software Engineering*, SE-7(2):223–228, 1981.

[27] C.-S. Shin, S. K. Kim, and K.-Y. Chwa. Area-efficient algorithms for straight-line tree drawings. *Computational Geometry*, 15(4):175–202, 2000.

[28] L. Trevisan. A note on minimum-area upward drawing of complete and Fibonacci trees. *Information Processing Letters*, 57(5):231–236, 1996.

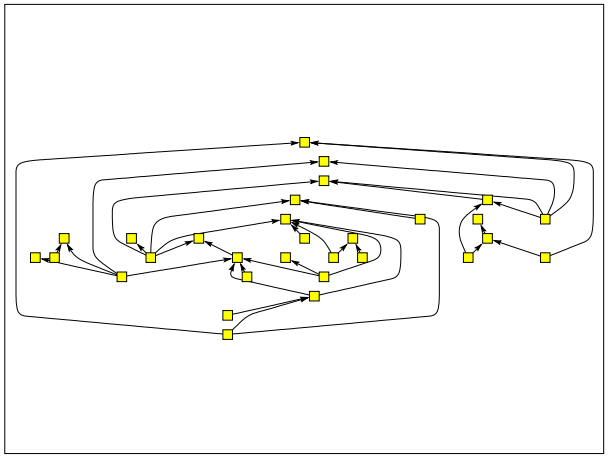


(a)

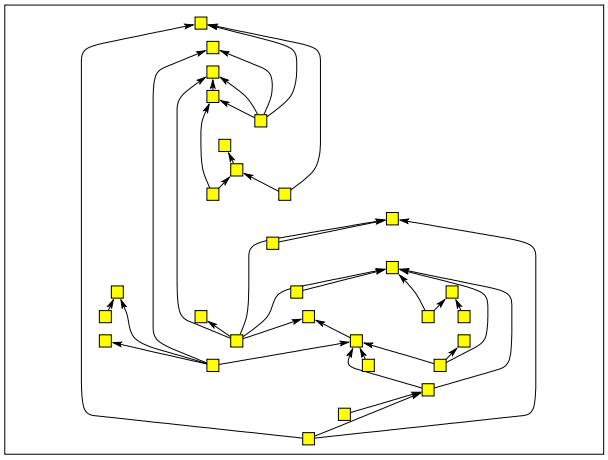


(b)

Figure 12: Two upward drawings of the same digraph with 10 vertices: (a) `SimpleSat`; (b) `SrSat`.

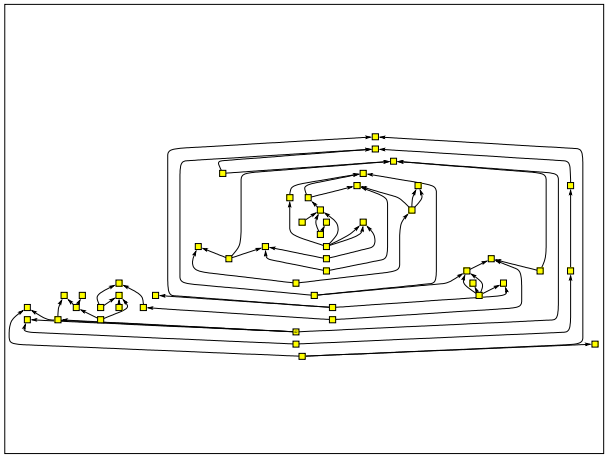


(a)

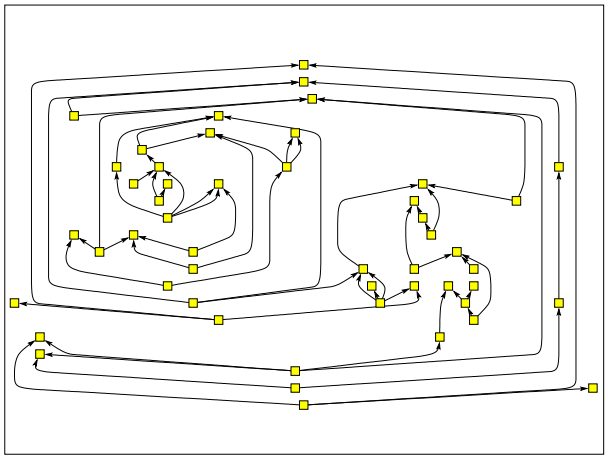


(b)

Figure 13: Two upward drawings of the same digraph with 30 vertices: (a) `SimpleSat`; (b) `SrSat`.

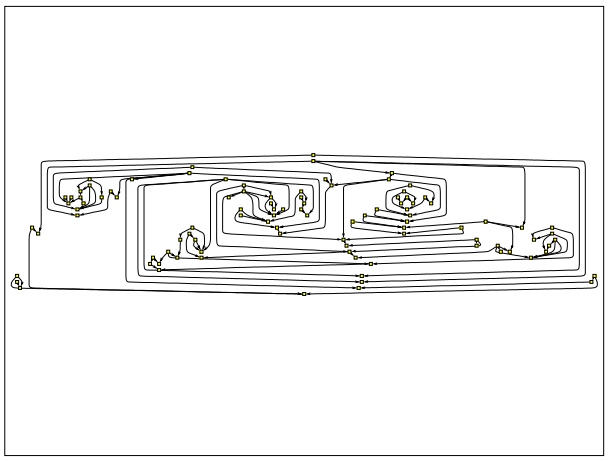


(a)

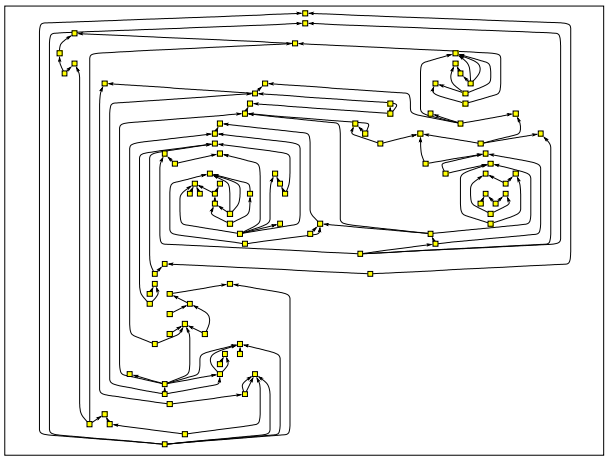


(b)

Figure 14: Two upward drawings of the same digraph with 50 vertices: (a) `SimpleSat`; (b) `SrSat`.



(a)



(b)

Figure 15: Two upward drawings of the same digraph with 100 vertices: (a) `SimpleSat`; (b) `SrSat`.

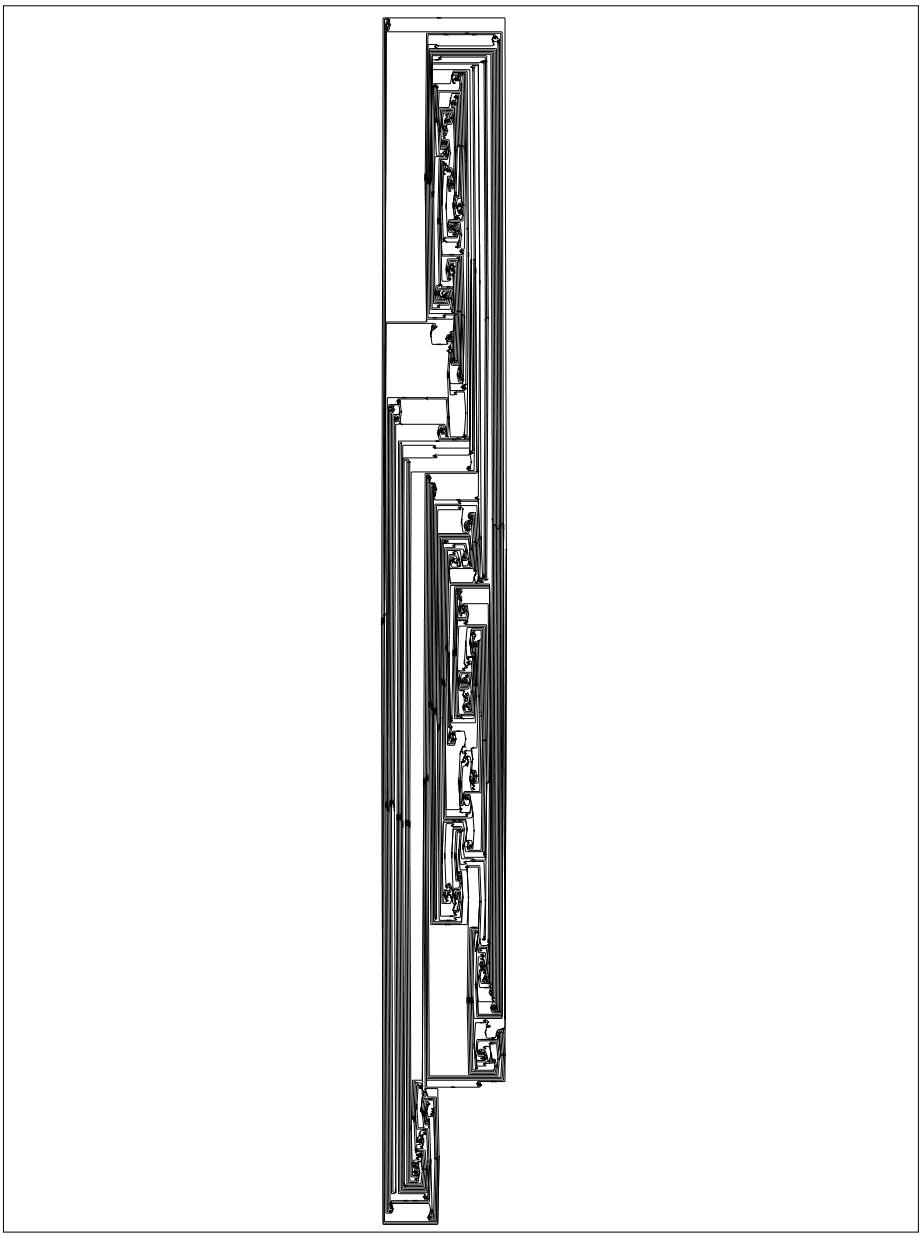


Figure 16: An upward drawing (in landscape) of a digraph with 1000 vertices, computed with heuristic `SimpleSat`. The deviation from a 4/3 screen is very high.

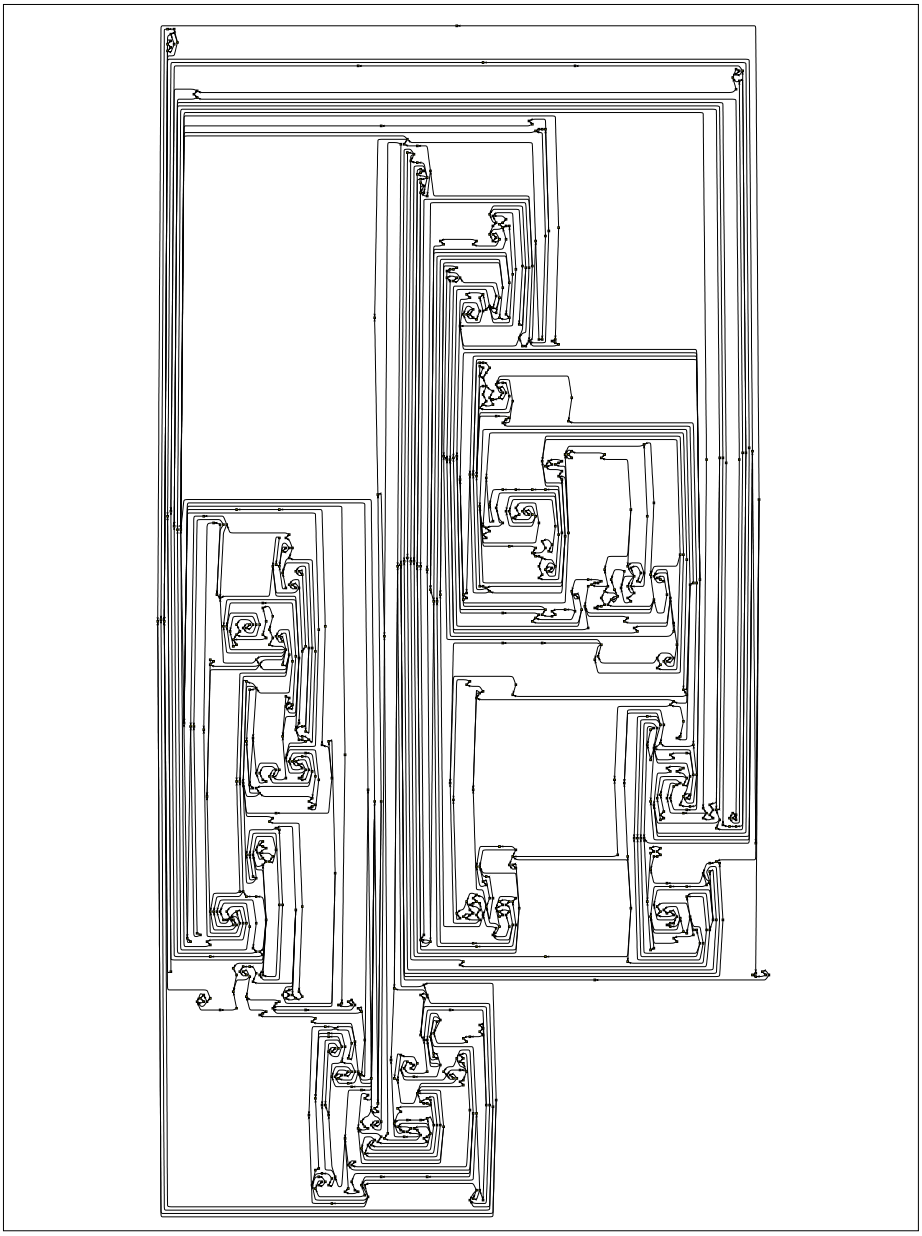


Figure 17: An upward drawing (in landscape) of the digraph in Figure 16, computed with heuristic `SrSat`. The aspect ratio is quite better.