Overloaded Orthogonal Drawings

Evgenios M. Kornaropoulos[1,2] and Ioannis G. Tollis[1,2]

[1] Department of Computer Science, University of Crete, Heraklion, Crete, Greece
[2] Institute of Computer Science, Foundation for Research and Technology-Hellas, Vassilika Vouton, P.O. Box 1385, Heraklion, GR-71110 Greece
{kornarop,tollis}@ics.forth.gr

Abstract. Orthogonal drawings are widely used for graph visualization due to their high clarity of representation. In this paper we present a technique called Overloaded Orthogonal Drawing. We first place the vertices on grid points following a relaxed version of dominance drawing, called weak dominance condition. Edge routing is implied automatically by the vertex coordinates. In order to simplify these drawings we use an overloading technique. All algorithms are simple and easy to implement and can be applied to directed acyclic graphs, planar, non-planar and also undirected graphs. We also present bounds on the number of bends and the area. Overloaded Orthogonal drawings present several interesting properties such as efficient visual edge confirmation as well as simplicity and clarity of the drawing.

1 Introduction

An *orthogonal drawing* maps each edge into a chain of horizontal and vertical line segments. An *orthogonal grid drawing* is an orthogonal drawing such that vertices and bends along the edges have integer coordinates. Drawings in this style are useful in many applications due to the high clarity of the model. The problem of constructing an orthogonal drawing while minimizing several aesthetic criteria such as area, bends, maximum edge length and total edge length is an NP-hard problem [4]. Therefore most algorithms employ heuristics that try to layout the graph in a manner which is good for some set of aesthetics.

Various algorithms have been introduced to produce orthogonal drawings of planar graphs [18,2,20,19,4]. A necessary and sufficient condition for a plane graph with maximum degree three to have an orthogonal drawing without bends was presented in [17]. Another interesting result is that an outerplanar graph G with maximum degree at most three has an orthogonal drawing with no bends if and only if G contains no triangles [12]. Bertolazzi et al. presented [1] a branch and bound algorithm that computes an orthogonal representation with the minimum number of bends of a biconnected planar graph. For drawings of non-planar graphs [9,3,13], the required area can be as little as $0.76n^2$ [14], the total number of bends is no more than $2n + 2$ [2,14], and each edge has at most two bends. Experimental studies have been conducted where various proposed algorithms were tested on their performance on area, bends, crossings, edge length, and time [21]. Dominance drawings are a widely used technique for visualizing planar *st*-graphs. These drawings have numerous useful features such as, small number of bends, small area, linear time complexity, detection and display of symmetries [4,5].

In this paper we introduce the overloaded orthogonal model which combines dominance and row/column reuse. We use a concept of relaxed dominance for vertex coordinate assignment, and orthogonal grid layout with overloaded use of rows/columns for edge routing. This type of routing has been used extensively in VLSI layout [11]. The concept of merging together groups of edges has been also used in the confluent drawing framework [6,7] to facilitate readability of the graph. This model can be applied to both planar and non-planar graphs. Also it can be efficiently applied to graphs with maximum degree four, and to graphs with degree higher than four. The presented algorithms produce drawings with at most $n - 1$ bends, $O(n^2)$ area, they run in linear time $O(n + m)$, and are easy to implement. Although a direct comparison with the bounds of traditional orthogonal drawings is a bit unfair (due to the reuse of rows and columns) our bounds on the number of bends and area are promising. Furthermore, every overloaded orthogonal drawing simplifies tremendously the visual confirmation of the existence of an edge and/or path between any two vertices.

This paper is organized as follows: in Section 2 we present an algorithm for constructing overloaded orthogonal drawings. In Section 3 we discuss some properties of the proposed model. In Section 4 we present properties and bounds of the overloaded orthogonal model in directed acyclic graphs. Section 5 gives an application of the proposed model to other graphs and finally Section 6 gives conclusions and open problems.

2 Overloaded Orthogonal Framework

In this framework, we propose to place the vertices in the grid so that edges flow from left-to-right and from bottom-to-top. Each vertex u is placed on a point in the grid with coordinates $X(u)$ and $Y(u)$. Dominance drawings achieve this vertex placement for *st*-planar graphs. A dominance drawing Γ of a graph $G = (V, E)$ has the following property: for any two vertices $u, v \in V$ there is a directed path from u to v in G, if and only if $X(u) \leq X(v)$ and $Y(u) \leq Y(v)$ in Γ. But, not every directed acyclic graph has a dominance drawing. Therefore we propose a relaxed condition, called *weak dominance condition*, that can be applied to any directed acyclic graph (dag):

Weak Dominance Condition: Let $G = (V, E)$ be a directed acyclic graph. For any two vertices $u, v \in V$ if there is a directed path from u to v in G, then $X(u) \leq X(v)$ and $Y(u) \leq Y(v)$.

Thus if v is in the upper-right quadrant of u, then v is not necessarily reachable from u. A path that is implied by the vertex coordinates but does not exist in G is called a *falsely implied path* (or *fip*). The problem of minimizing the number of falsely implied paths was introduced in [10], where it is shown that the corresponding decision problem is NP-complete.

Following the footsteps of the algorithm for dominance drawing for (reduced) planar *st*-graphs presented in [5], we formulate an algorithm for vertex placement that respects the weak dominance condition and is applicable to any dag. The main algorithm for planar *st*-graphs described in [5] consists of three phases. In the first phase, called 'Preprocessing Phase', a linked data structure is constructed in order to efficiently calculate coordinates. During the second phase called 'Preliminary Layout' distinct X, Y coordinates are given to each vertex. In the third and final phase, a compaction procedure is applied to reduce the area of the drawing.

We will construct a similar data structure as in 'Preprocessing step', but for general directed acyclic graphs. Let W be a representation of a dag G such that the incoming edges for each vertex u appear consecutively around u. Representation W will be called a *representation in consecutive form.* The representation in consecutive form is a method to force a left-to-right order in the incoming as well as outgoing edges of every vertex of G. Without loss of generality we assume that there is only one source, s. If not then we insert an artificial super-source s and connect it to all sources of G. The algorithm performs two topological sortings on the vertices of G. Successors of each vertex are scanned in clockwise order for the X coordinate assignment, and in counterclockwise order for the Y coordinate assignment. The order is imposed according to the representation in consecutive form that is given as an input. We will present the algorithm for clockwise scan, that computes the X-coordinate assignment.

```
Algorithm. TOPOLOGICAL-SORTING(Adj(G))
1. for each vertex $v \in$ V
2.       X[$v$]$\leftarrow \infty$
3. X[$s$]$\leftarrow$ 0
4. time$\leftarrow$1
5. VISIT-CLCK($s$)
6. return X

Algorithm. VISIT-CLCK($u$)
1. for each vertex $v \in$ Adj($u$) such that $(u, v)$
             is the leftmost outgoing edge of $u$ do
2.       if in-degree($v$)=1
3.             X[$v$]$\leftarrow$ time
4.             time$\leftarrow$time+1
5.             remove edge $e$=$(u, v)$
6.             VISIT-CLCK ($v$)
7.       else
8.             remove edge $e$=$(u, v)$
```

Algorithm TOPOLOGICAL-SORTING scans the outgoing edges of a vertex u in clockwise order (leftmost outgoing edge) and visits a direct successor v only if v has in-degree one. Otherwise, it removes edge (u, v) from the list. Analogously, we formulate an algorithm for the Y-coordinate assignment that performs a counterclockwise scan, by replacing VISIT-CLCK with VISIT-COCLCK. The difference between the two VISIT algorithms is Line 1, where instead of leftmost outgoing edge we now have rightmost outgoing edge. The two topological sortings are used by WDP algorithm for assigning X and Y coordinates to the vertices of G.

```
Algorithm. (WDP)WEAK DOMINANCE PLACEMENT ($W$)
1. X coordinates ← TOPOLOGICAL SORTING($W$) using VISIT-CLCK
2. Y coordinates ← TOPOLOGICAL SORTING($W$) using VISIT-COCLCK
```

We denote the number of vertices in G by n, and the number of edges in G by m. Since both topological sorting algorithms run in linear time $O(n + m)$, algorithm WDP also runs in linear time $O(n + m)$.

In the rest of this section we will see how the Algorithm WDP creates a natural separation between *pq*-components. A *pq*-component $G_{pq} = (V', E')$ of G is a maximally induced subgraph of G with a single source p and a single sink q that contains at least two edges and that is connected with the rest of G only through vertex p and vertex q. Thus, vertex p is a dominator of every vertex $v \in V'$ and q is a post-dominator of every vertex $v \in V'$. Due to space limitations, the proofs of the following results are omitted.

Lemma 1. *If dag G=(V,E) includes a pq-component* $G_{pq} = (V', E')$*, then* $X(q) = X(p) + |V'| - 1$ *and* $Y(q) = Y(p) + |V'| - 1$.

Corollary 1. *If dag G=(V,E) includes a pq-component* $G' = (V', E')$ *, then for every vertex* $u \in G'$, $X(p) \leq X(u) \leq X(p) + |V'| - 1$ *and* $Y(p) \leq Y(u) \leq Y(p) + |V'| - 1$.

Let $X()$ and $Y()$ be the coordinates constructed by WDP algorithm. Also let $G' = (V', E')$ be a component where $V' \subseteq V$ and $E' \subseteq E$. A component G' is said to be *separated*, if the following property holds for $X()$ and $Y()$:

$$\forall u \in V', v \in V - V' \Rightarrow (X(u) \leq X(v) \wedge Y(u) \leq Y(v)) \vee (X(u) \geq X(v) \wedge Y(u) \geq Y(v))$$

This property is a guarantee that every vertex $v \in V - V'$ that is not a member of a component G' will not appear between the vertices of G'. We refer to this as the *separation property.*

Theorem 1. *Vertex placement* $X()$ *and* $Y()$ *constructed by algorithm WEAK DOMINANCE PLACEMENT respects the separation property for every pq-component.*

Proof. (Sketch) Let $G' = (V', E') \subseteq G$ be a *pq*-component. Then algorithm TOPOLOGICAL - SORTING for G, returns a numbering of vertices of G' from $X(p)$ to $X(p) + |V'|$. Also holds for Y-coordinates, i.e., numbers vertices of G' from $Y(p)$ to $Y(p) + |V'|$. Thus, no vertex from $V - V'$ can be drawn inside a *pq*-component. □

Lemma 2. *Let u and v be a pair of vertices of G such that $X(v) = X(u) + 1$. Then $Y(u) < Y(v)$ if and only if G has an edge (u,v).*

Lemma 3. *Let u and v be a pair of vertices of G such that $Y(v) = Y(u) + 1$. Then $X(u) < X(v)$ if and only if G has an edge (u,v).*

Our proposed framework contains the term 'overloaded' because all outgoing edges of a vertex use the same column in order to reach their corresponding destination vertex. We will first discuss how a single edge is routed, and then we will focus on unambiguously visualizing the edges of the drawing.

Edge routing is automatically implied by the coordinates of the vertices. Each edge (u, v) consists of a vertical edge segment from $(X(u),Y(u))$ to $(X(u),Y(v))$ and a horizontal segment from $(X(u),Y(v))$ to $(X(v),Y(v))$. Because various edges reuse segments of rows and columns we introduce e-points to resolve ambiguities, see Figure 1. Given an edge (u, v) an *e-point* is defined as an unlabeled point that is placed on point $(X(u), Y(v))$ to indicate a direct connection from u to v. A bend will appear in the final drawing instead of an e-point if: (a) vertex u does not have a successor w such that $Y(w) \geq Y(v)$ and (b) vertex v does not have a predecessor z such that $X(z) \leq X(v)$.

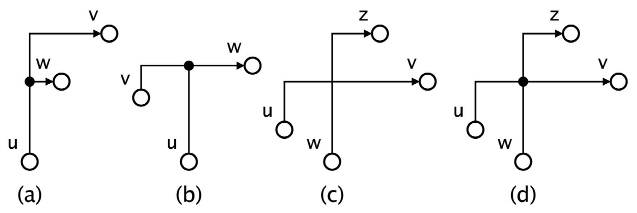

Fig. 1. (a) the vertical segment of (u, v) is overloaded by the vertical segment of (u, w). To visualize the edge from u to w, an e-point is placed at $(X(u), Y(w))$. (b) the horizontal segment of (v, w) is overloaded by the horizontal segment of (u, w). To visualize the edge from u to w, an e-point is placed at $(X(u), Y(w))$. If there is no e-point then $(w, v) \notin E$ (c), whereas if there is an e-point in $(X(w),Y(v))$ then $(w, v) \in E$ (d).

We will describe an algorithm that receives the vertex coordinates as an input, and outputs an overloaded orthogonal drawing. It routes the edges according to the given coordinates and places e-points where needed.

In order to construct an overloaded orthogonal drawing a linked data structure for G will be constructed. Each vertex $u \in V$ of G, points to the list of its direct successors sorted in decreasing order according to their Y-coordinate. This single linked list of u, can be traversed by means of pointer *next(u)*. It can also be accessed by pointer *getFirst(u)*, that is u's direct successor with the highest Y-coordinate (hence first in the list). In case of a tie, we can arbitrarily order vertices with the same coordinate without affecting the overall result.

```
Algorithm. (OOD) OVERLOADED ORTH. DRAWING(Adj(G) , X() ,Y() )
1. for each vertex $u \in$ V
2.         visited[$u$]$\leftarrow$ 0
3. for each vertex $u \in V$ in increasing order of X-coordinate
4.         $v \leftarrow$ next($u$)
5.         while $v \neq nil$
6.                 Draw edge segment from (X($u$),Y($u$)) to (X($u$),Y($v$))
7.                 Draw edge segment from (X($u$),Y($v$)) to (X($v$),Y($v$))
8.                 if getFirst($u$)$\neq v$ OR visited[$v$]$\neq$0
9.                         New $e$-point $\leftarrow$ ( X($u$),Y($v$) )
10.                visited[$v$] $\leftarrow$ 1
11.                $v \leftarrow$ next($v$)
12. end
```

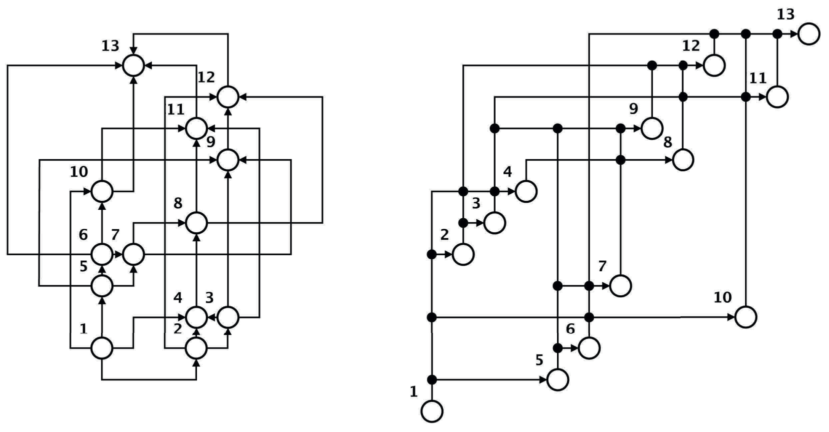

Fig. 2. Two different drawings of a regular degree four graph with 13 vertices and 26 edges. In the left picture an orthogonal grid drawing is depicted, the graph and the drawing are taken from [4]. While, in the right picture there is an overloaded orthogonal drawing of the same graph. No compaction was performed to the overloaded orthogonal drawing.

Theorem 2. *Algorithm OOD produces an overloaded orthogonal drawing Γ of G with vertex coordinates computed by algorithm WDP. Γ has at most $n - 1$ bends, $O(n^2)$ area and is constructed in $O(n + m)$ time.*

2.1 Compaction

Compaction is applied as a post-processing step in an overloaded orthogonal drawing in order to reduce the X- and Y-coordinates. Our compaction follows the steps of the Algorithm in [4,5]. However since our graphs are not planar, and therefore we do not have planar embeddings, we need to be extra careful in order to produce a valid drawing. In this step we allow equality between vertex coordinates under the following conditions: (a) The compaction is performed between vertices $u, v \in V$ such that there is an edge $(u, v) \in E$. (b) Two distinct vertices cannot coincide in the same point. (c) Compaction on the X- or Y-coordinates will not be performed if an edge is forced to pass over u or any other vertex.

3 Clarity and Readability of the Model

In this section we outline some advantages of the overloaded orthogonal model.

• *Meaningful relation between vertex coordinates*: The weak dominance condition implies that: if there is a path from u to v then vertex v will appear in the upper right quadrant of vertex u.

• *Works for any pair of topological sortings as X, Y coordinates*: Since every pair of topological sortings respects the weak dominance condition, we can take any pair of topological sorting as X, Y coordinates.

• *Universality of the model*: The overloaded orthogonal model does not discriminate between graphs with maximum degree four, and graphs with higher degree. Furthermore, it can be efficiently applied to planar and to non-planar graphs. The overloaded orthogonal model can also be applied to undirected graphs, given that an *st*-numbering with various properties can be computed for any undirected graph [15,16] . An interesting example is presented in Section 5.

• *Efficient Visual Confirmation of an Edge*: We can visually confirm the existence of an edge (u, v) by checking if there is an e-point or a bend on point $(X(u), Y(v))$. If a compaction is performed u or v could replace the e-point at the location $(X(u), Y(v))$. In contrast, in the regular orthogonal model we would visually follow every outgoing edge of u successively, until we reach v. Consequently, the size of a graph does not affect the readability of an overloaded orthogonal drawing, as we can check if any two vertices are connected by inspecting only a single point i.e., in $O(1)$ time.

• *Efficient Visual Confirmation of Reachability*: An interesting extension of this graph drawing technique occurs when we use the transitive closure of a graph as input. In that case every possible path along the original directed acyclic graph $G = (V, E)$ will be represented by an edge in the transitive closure $G^* = (V, E^*)$. By applying the overloaded orthogonal model we can check if a vertex v is reachable from a vertex u by examining point $(X(u), Y(v))$ in the drawing. As shown in Figure 3, e-points of the corresponding transitive edges are colored grey. Notice that there is no e-point at $(X(4), Y(9))$, despite the fact that the coordinates of vertex 9 dominate the coordinates of vertex 4. In this context, crossings indicate the existence of falsely implied paths.

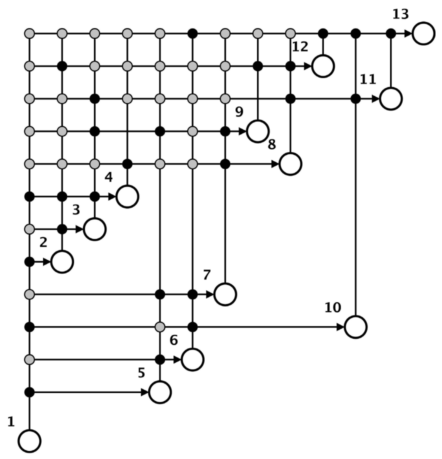

Fig. 3. An overloaded orthogonal drawing of the transitive closure. Reachability of any pair of vertices u-v can be confirmed by looking at point $(X(u), Y(v))$. By the the color of the e-point we can determine if there is an edge or a path between the vertices.

4 Directed Acyclic Graphs

In this section we present several properties and bounds of overloaded orthogonal drawings for directed acyclic graphs. If $X(u) \neq X(v)$ and $Y(u) \neq Y(v)$ for every pair of vertices $u, v \in V$, then every edge has a 'step'-like form and consequently produces either a bend or an e-point. Therefore we have:

Lemma 4. *Let Γ be an overloaded orthogonal drawing of dag G, where each vertex is placed in a distinct X, Y coordinate. Then $bends(\Gamma)+ePoints(\Gamma) = m$.*

If a compaction is performed on drawing Γ, then the sum $bends(\Gamma)+ePoints(\Gamma)$ would be less than the number of edges. Additionally, every vertex can have at most one bend on its row. That bend is produced from its direct predecessor with the lowest X-coordinate. Taking into consideration that sources do not have incoming edges, we have the following lemma:

Lemma 5. *Let Γ be any overloaded orthogonal drawing of a dag G. Let also n_s be the number of sources of G. Then $bends(\Gamma) \leq n - n_s$.*

The upper bound of the above lemma is tight as shown by the following theorem.

Theorem 3. *There exists a family of planar n-vertex graphs G_n, for $n \geq 3$, such that any overloaded orthogonal drawing Γ of G_n requires at least $n-2$ bends, and $(n-2)\times(n-2)$ area.*

Proof. (Sketch) Consider the graph G_n shown in Figure 4. Each vertex u_i has two outgoing edges, (u_i, u_{i+2}) and (u_i, u_{i+1}). The transitive closure of this family of

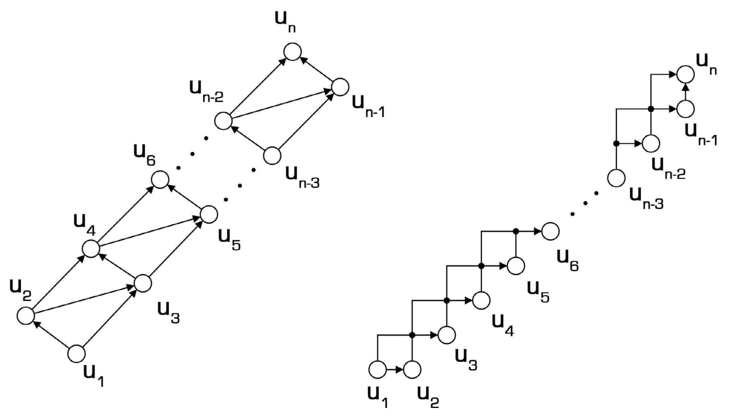

Fig. 4. An explanatory construction of Theorem 3

graphs is a complete directed acyclic graph, therefore the topological sorting for this graph is unique. Their drawings admit a single compaction in Y-coordinate between vertex u_1 and vertex u_2, and a single compaction in X-coordinate between vertex u_{n-1} and vertex u_n. Therefore an overloaded orthogonal drawing of this family of graphs has optimal area $(n-2)\times(n-2)$, and has at least $n-2$ bends. □

The dominance drawing technique was applied to reduced planar st-graphs in [5]. If we apply the edge routing technique using the vertex coordinates produced by the dominance drawing algorithm presented in [5], the drawing has zero bends.

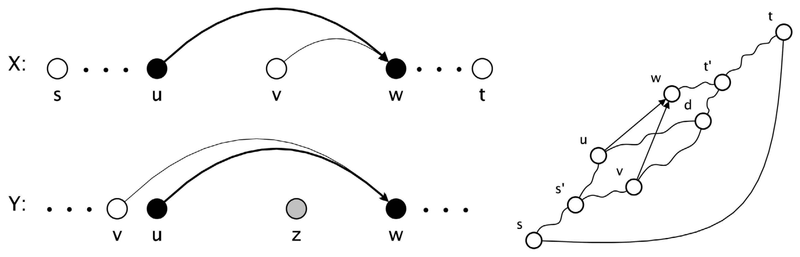

Fig. 5. Proof of Theorem 4. The left picture illustrates the difference between ($z1$)-case and ($z2$)-case. In the right picture there is a drawing of a $K_{3,3}$ that exists in ($z2$)-case.

Theorem 4. *Given a reduced planar st-graph $G = (V, E)$, an overloaded orthogonal drawing Γ with zero bends can be constructed in linear time, $O(n)$.*

Proof. (Sketch) Consider a reduced planar st-graph G with vertex coordinates obtained by the dominance drawing algorithm in [5]. Let an edge $(u, w) \in E$ such that it forms a bend that cannot be removed by a compaction. We construct such a scenario and prove that this edge cannot exist without contradicting the basic assumptions. Vertex u and vertex v cannot be consecutive in X-coordinate. Thus there must be a vertex v such that $X(u) < X(v) \leq X(w)$. Let also a vertex z such that $Y(u) < Y(z) < Y(w)$. Vertex z cannot be between u and w in X-coordinate due to the fact that G is reduced. Thus, we have two different cases: $(z1)$ where $X(s) < X(z) < X(u)$ and $(z2)$ where $X(w) < X(z) < X(t)$. Case $(z1)$ will conclude that edge (u, w) is transitive. Case $(z2)$ will conclude that there is a graph homeomorphic to $K_{3,3}$ and consequently G is not planar, a contradiction in both cases. □

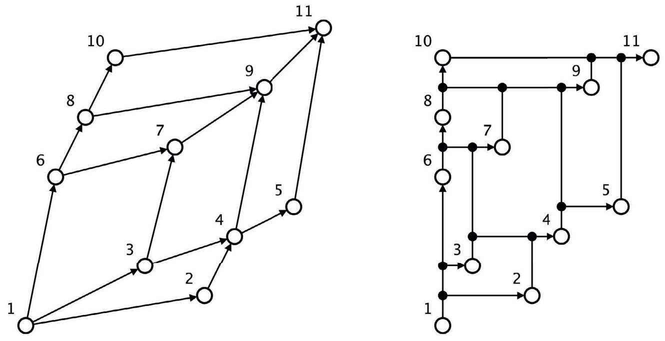

Fig. 6. In the left figure we have the straight-line dominance drawing of a reduced planar st-graph as described in [4]. In the right figure there is a compacted overloaded orthogonal drawing of the same graph with zero bends.

5 Other Graphs

In this section the overloaded orthogonal model is going to be extended to draw undirected graphs and directed graphs with cycles. Let G be an undirected graph and s, t be two distinct vertices of G. If the graph is planar we first construct a planar embedding and proceed, otherwise we ignore that step. An st-numbering for G is a numbering $v_1, v_2, \ldots, v_n$ of the vertices of G such that $s = v_1$, $t = v_n$, and every vertex v_j, other than s and t, is adjacent to at least two vertices v_i and v_k with $i < j < k$. Such a numbering can be constructed in linear time [8]. Given an st-numbering we orient the edges of E from the low-numbered vertex to the high numbered one. We name the resulting digraph D. The algorithm for st-orientation proposed in [15,16], parametrically controls the length of the longest path of the final st-oriented graph. As it was expected, different values of parameter p yield overloaded orthogonal drawings with different characteristics. We can apply the vertex placement algorithm to D, and then route the edges as described in Algorithm OOD. A compaction step can also be performed. As shown in Figure 7, the st-orientation with $p = 0$ results in an overloaded orthogonal drawing with area 19×19, while the st-orientation with $p = 1$ results in an overloaded orthogonal drawing with optimal area 2×19. We are conducting an experimental study in order to investigate the influence of an st-numbering of G, on the area of its overloaded orthogonal drawing Γ.

If G is a directed graph with cycles one could find a minimal feedback arc set F [4] and obtain an uncompacted overloaded orthogonal drawing of $G - F$. Complete the drawing by routing each edge $(u, v) \in F$ as follows: vertical segment from $(X(u), Y(u))$ to $(X(u), Y(v))$, horizontal segment from $(X(u), Y(v))$ to $(X(v), Y(v))$, placing e-points where necessary. Notice that rows and columns used for routing these edges, have not been used to route the edges of $G - F$.

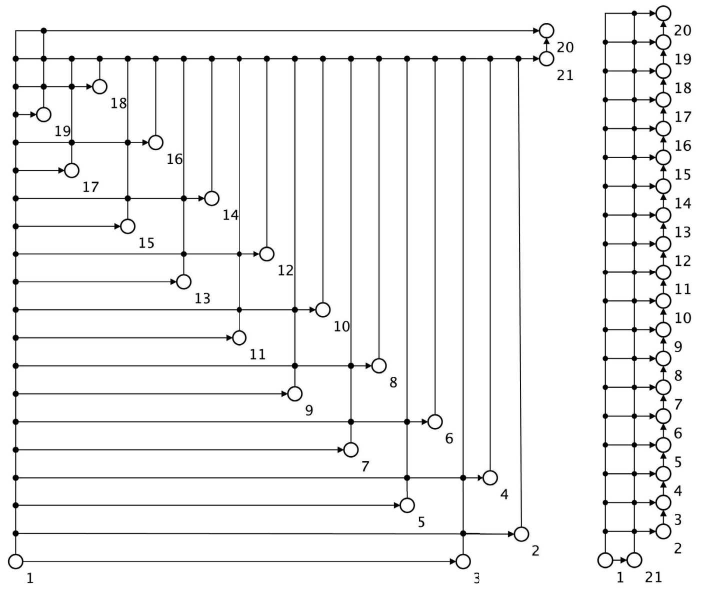

Fig. 7. Two overloaded orthogonal drawings of an originally undirected planar graph are shown. Left: the st-orientation was produced by algorithm [15] with parameter $p = 0$, right: same algorithm with parameter $p = 1$.

6 Conclusion and Open Problems

We presented algorithms that produce overloaded orthogonal drawings with at most $n - 1$ bends, $O(n^2)$ area, they run in linear time $O(n + m)$, and are easy to implement. An interesting open problem is to find algorithms for weak dominance placement that provide upper bounds on the number of crossings in an overloaded orthogonal drawing of the transitive closure.

References

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

2. Biedl, T., Kant, G.: A better heuristic for orthogonal graph drawings. Computational Geometry: Theory and Applications 9(3), 159–180 (1998)
3. Biedl, T.C., Madden, B.P., Tollis, I.G.: The Three-Phase Method: A Unified Approach to Orthogonal Graph Drawing. Int. J. Comput. Geometry Appl. 10(6), 553–580 (2000)
4. Di Battista, G., Eades, P., Tamassia, R., Tollis, I.G.: Graph Drawing: Algorithms for the Visualization of graphs. Prentice - Hall, New Jersey (1998)
5. Di Battista, G., Tamassia, R., Tollis, I.G.: Area Requirement and Symmetry Display of Planar Upward Drawings. Discrete and Comput. Geom. 7(4), 381–401 (1992)
6. Dickerson, M., Eppstein, D., Goodrich, M.T., Meng, J.Y.: Confluent Drawings: Vizualizing Non-planar Diagrams in a Planar Way. Journal of Graph Algorithms and Applications 9(1), 31–52 (2005)
7. Eppstein, D., Goodrich, M.T., Meng, J.Y.: Confluent Layered Drawings. Algorithmica 47(4), 439–452 (2007)
8. Even, S., Tarjan, R.: Computing an st-numbering. Theoretical Computer Science 2(3), 339–344 (1976)
9. Fößmeier, U., Kaufmann, M.: Algorithms and Area Bounds for Nonplanar Orthogonal Drawings. In: DiBattista, G. (ed.) GD 1997. LNCS, vol. 1353, pp. 134–145. Springer, Heidelberg (1997)
10. Kornaropoulos, E.M., Tollis, I.G.: Weak Dominance Drawings and Linear Extension Diameter, arXiv:1108.1439 (2011)
11. Lengauer, T.: Combinatorial algorithms for integrated circuit layout. John Wiley & Sons, Inc., New York (1990)
12. Nomura, K., Tayu, S., Ueno, S.: On the Orthogonal Drawing of Outerplanar Graphs. Journal IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences E88-A(6), 1583–1588 (2005)
13. Papakostas, A., Tollis, I.G.: Efficient Orthogonal Drawings of High Degree Graphs. Algorithmica 26(1), 100–125 (2000)
14. Papakostas, A., Tollis, I.G.: Algorithms for Area-Efficient Orthogonal Drawings. Computational Geometry Theory and Applications 9(1-2), 83–110 (1998)
15. Papamanthou, C., Tollis, I.G.: Algorithms for computing a parameterized st-orientation. Theoretical Computer Science 408(2-3), 224–240 (2008)
16. Papamanthou, C., Tollis, I.G.: Applications of Parameterized st-Orientations. Journal of Graph Algorithms and Applications 14(2), 337–365 (2010)
17. Rahman, S., Nishizeki, T., Naznin, M.: Orthogonal Drawings of Plane Graphs Without Bends. Journal of Graph Algorithms and Applications 7(4), 335–362 (2003)
18. Storer, J.: On minimal node-cost planar embeddings. Networks 14(2), 181–212 (1984)
19. Tamassia, R.: On embedding a graph in the grid with the minimum number of bends. SIAM J. Computing 16(3), 421–444 (1987)
20. Tamassia, R., Tollis, I.G.: Planar Grid Embeddings in Linear Time. IEEE Transactions on Circuits and Systems 36(9), 1230–1234 (1989)
21. Vismara, L., Di Battista, G., Garg, A., Liotta, G., Tamassia, R., Vargiu, F.: Experimental studies on graph drawing algorithms. Software: Practice and Experience 30(11), 1235–1284 (2000)