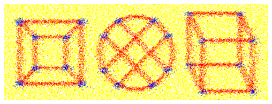
Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 20, no. 2, pp. 217–246 (2016)
DOI: 10.7155/jgaa.00391

Algorithms and Bounds for Overloaded Orthogonal Drawings

Evgenios M. Kornaropoulos[1] *Ioannis G. Tollis*[2]

[1]Computer Science Department, Brown University, Providence, RI, USA
[2]Computer Science Department, University of Crete, Heraklion, Crete, Greece

Abstract

Orthogonal drawings are widely used for graph visualization due to their high clarity and ease of representation. But when it comes to high-degree and massive graphs, even orthogonal drawings have difficulties producing a clean and simple visualization. In this paper we present a technique called *Overloaded Orthogonal Drawing* that greatly improves the readability by proposing a new vertex placement and reframing the orthogonal edge routing approach. 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 of the edges. All algorithms are simple and easy to implement and can be applied to directed acyclic graphs, planar, non-planar and 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 clarity of the drawing.

Submitted:	Reviewed:	Revised:	Accepted:	Final:
December 2014	May 2015	October 2015	December 2015	January 2016

Published: February 2016

Article type: Regular paper

Communicated by: G. Liotta

E-mail addresses: evgenios@cs.brown.edu (Evgenios M. Kornaropoulos) tollis@csd.uoc.gr (Ioannis G. Tollis)

1 Introduction

The visualization of graphs is important in many application areas in computer science, telecommunications, biology, social networks, finance, transportation, etc. In order to produce nice and readable drawings various aesthetic criteria have been devised. From a human readability point of view, *orthogonal drawings*, where each edge is drawn as a polygonal chain of alternating horizontal and vertical segments, have received a lot of attention, see for example [6]. Unfortunately, due to the fact that a 2-dimensional grid-point is adjacent to four half-lines, most techniques for orthogonal drawings limit the maximum degree of the vertices of the input graph to four [3, 36, 37, 38]. In this paper we introduce a new model, called *overloaded orthogonal drawing*, where vertices are represented by points and edges are allowed to overlap in their corresponding rows and/or columns.

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 problems of constructing an orthogonal drawing while minimizing several aesthetic criteria such as area, bends, maximum edge length and total edge length are NP-hard [6]. 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 presented to produce orthogonal grid drawings of planar and non-planar graphs of maximum degree four [3, 6, 13, 29, 36, 37, 38]. A necessary and sufficient condition for a plane graph with maximum degree three to have an orthogonal drawing without bends was presented in [33]. 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 [28]. Bertolazzi et al. presented [2] a branch and bound algorithm that computes an orthogonal representation with the minimum number of bends of a biconnected planar graph. For drawings of general (i.e., non-planar) undirected graphs of maximum degree four, the required area can be as little as $0.76n^2$ [29], the total number of bends is no more than $2n + 2$ [3, 29], 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 [39].

In order to extend the possibility of constructing orthogonal representations, some techniques draw the nodes as rectangles that occupy bigger area and hence they have several available grid points to attach to their incident edges. To name a few, the Kandinsky model [1, 5, 12, 13], the three-phase method [4] and others [30] propose heuristics to fit a high-degree undirected graph to the orthogonal drawing framework.

Besides the orthogonal drawing technique, the proposed new model of this work is inspired in part by the so-called *dominance drawings* which is a widely used technique for visualizing planar *st*-graphs (planar graphs with one source and one sink, see Section 2). The vertex placement in the dominance drawing technique assigns grid locations to nodes such that a vertex v has coordinates X

and Y greater than (or equal to) any other vertex u if and only if there exists a directed path from u to v. Dominance drawings have numerous useful features such as small number of bends, small area, linear-time complexity, detection and display of symmetries [6, 7].

In this paper we introduce the overloaded orthogonal drawing model for directed acyclic graphs (DAGs) which combines the useful characteristics of the dominance drawing technique with row and/or column reuse for the edges. We introduce the concept of relaxed dominance or *weak dominance* for vertex coordinate assignment, by placing a vertex v in the top right (northeast) quadrant defined by vertex u if there is a path from u to v. Notice that in the weak dominance placement the reverse cannot be guaranteed, i.e., a node w could be placed in the northeast quadrant defined by node u, even if there is no path from u to w. After the computation of such a placement the edges are naturally routed in the orthogonal grid by reusing rows and columns, which we call *overloaded use of rows/columns for edge routing*. Figure 1 shows a comparison between an orthogonal grid drawing and an overloaded orthogonal drawing of the same DAG.

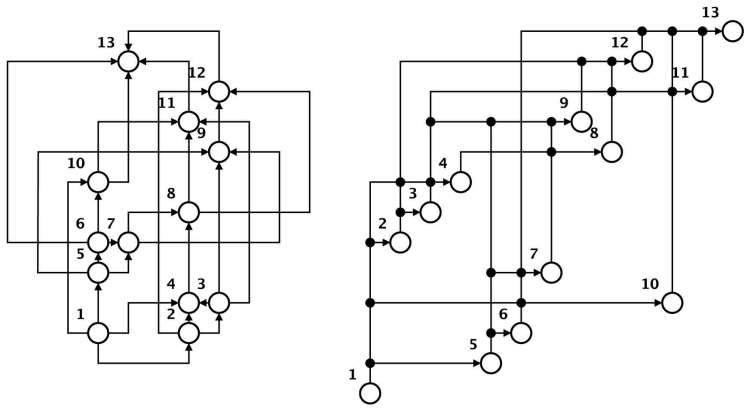

Figure 1: A directed acyclic graph with 13 nodes and 26 edges taken from [6] visualized using the orthogonal grid drawing (left) and the overloaded orthogonal technique (right). In the overloaded orthogonal model a directed edge from u to v corresponds to a 1-bend poly-line from the point $(X(u), Y(u))$ to the point $(X(v), Y(v))$ of the underlying grid. In order to point out the existence of an edge, if there is no bend, we place a small black dot called e-point at point $(X(u), Y(v))$.

This type of routing has been used extensively in VLSI layout [26]. The concept of merging together groups of edges has also been used in the confluent drawing framework [8, 10] in order to facilitate the readability of the graph drawing. Confluent drawings can also be applied to both planar and non-planar graphs without any maximum degree limitation. Another model that explores

the potential of simplifying the drawing by grouping the edges is the so-called *edge-bundling technique* [14, 17, 18, 25, 35]. The underlying idea in this approach is that edges merge into bundles on the common joint paths and fan out again when they reach their destination.

The presented algorithms for overloaded drawings produce drawings with at most $n-1$ bends, $O(n^2)$ area, they run in linear $O(n+m)$ time, where n is the number of vertices and m is the number of edges. Although a direct comparison with the bounds of traditional orthogonal drawings is a bit unfair (due to the reuse of rows and columns) the low number of required bends makes the resulting drawings very readable. Furthermore, if the given directed graph is acyclic, we can compute its transitive closure and we can use a point notation (i.e., *p*-point) in order to signify the existence of a path without creating significant extra load and confusion on the drawing, or altering the mental map of the user. Therefore, 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 some preliminary definitions. In Section 3 we present the weak dominance condition. In Section 4 we present the framework for constructing overloaded orthogonal drawings of directed acyclic graphs and discuss how to augment an overloaded orthogonal drawing in order to show its transitive closure. In Section 5 we prove some properties of the proposed model. In Section 6 we describe how to compact the resulting drawing. Section 7 describes how to apply the proposed model to undirected graphs, and directed graphs with cycles, and finally Section 8 gives conclusions and summarizes the properties of our framework.

2 Preliminaries

In this section we give some preliminary definitions that will be used later in the paper. A graph is *planar* if it can be drawn in a plane without any two edges crossing, except at common endpoints. A *planar st-graph* $G = (V, E)$ is a planar directed acyclic graph with exactly one source vertex s (i.e., in-degree zero) and exactly one sink t (i.e., out-degree zero) such that it can be embedded in the plane with both s and t on the boundary of the external face of the embedding of G. Let $G = (V, E)$ be a directed acyclic graph (DAG), then an edge $(u, v) \in E$ is *transitive* if there is another directed path in G from u to v. A DAG is said to be *reduced* if there are no transitive edges in E. A *topological sorting* of a DAG $G = (V, E)$ is a linear ordering of its vertices such that vertex u comes before v in the ordering if there exists a path from u to v in G. A *transitive closure* of a DAG $G = (V, E)$ is a DAG $G' = (V, E')$ such that for all $u, v \in V$ there is an edge in E' if and only if there is a path from u to v in G. We say that a graph $G'' = (V'', E'')$ is an *induced subgraph* of a graph $G = (V, E)$ if G'' is isomorphic to a graph whose vertex set V'' is a subset of the vertex set V, and whose edge set E'' consists of all the edges of G with both end vertices in V''. A *feedback arc set* of a directed graph $G = (V, E)$ is a subset A' of edges

of G such that removing A' from G results in a directed acyclic graph, that is $G' = (V, E - A')$ is a DAG. The work of Karp [15, 19] shows that finding a minimum feedback arc set in a directed graph is NP-hard. In this work we use heuristic algorithms, such as the Greedy-Cycle-Removal algorithm [6], in order to compute a minimal feedback arc set.

Let $G = (V, E)$ be a DAG with a single source $s \in V$. We say that vertex $w \in V$ *dominates* a vertex $v \in V$ if every path from s to v passes through w. The dominance relation in G can be represented in compact form as a tree T, called *the dominator tree* of G, in which the dominators of a vertex v are its ancestors. Vertex w is the *immediate dominator* [16] of v if w is the parent of v in T. The *post-dominators* of G are defined as the dominators in the graph obtained from G by reversing all directed edges and assuming that all vertices are reachable from a (possibly artificial sink) vertex t.

A *rotation system* [27] of a directed graph G is a mapping ϕ that assigns to each vertex v a cyclic permutation ϕ_v of the directed edges incident to v. The cyclic permutation ϕ_v is also called the *rotation at vertex* v.

3 Weak Dominance

In this framework, we propose to place the vertices of a graph in the grid so that edges flow from bottom-to-top and from left-to-right. Each vertex u is placed on a point in the grid with coordinates $X(u)$ and $Y(u)$. A *dominance drawing* Γ of a planar *st*-graph G is a straight-line drawing, such that for any two vertices u and 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. A technique that produces dominance drawings of planar *st*-graphs can be found in [6, 7].

Dominance drawings have many important aesthetic properties, including small number of bends, good vertex placement, and symmetry display [7]. More interestingly, this technique encapsulates the aspect of characterizing the transitive closure of the digraph by means of a geometric dominance relation among the vertices. It would be useful to obtain such drawings not only for planar *st*-graphs but also for general directed acyclic graphs.

So, naturally the question arises: can we produce a similar dominance drawing Γ of any directed acyclic graph G, if it is not planar? That is, can we compute X, Y coordinates for a (non-planar) directed acyclic graph G such that for any two vertices u and v of G 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. Unfortunately, not every directed acyclic graph admits a dominance drawing. A simple graph that does not admit a dominance drawing is the "crown graph" with 6 vertices, depicted in Figure 2.

The notion of *dominance drawing dimension* of a directed acyclic graph explains the fundamental limitations of the problem. The dimension of a directed acyclic graph G is defined as the value of the smallest k for which a k-dimensional dominance drawing of G can be obtained [9]. The family of planar *st*-graphs is a subclass of directed acyclic graphs that have dominance drawing dimension

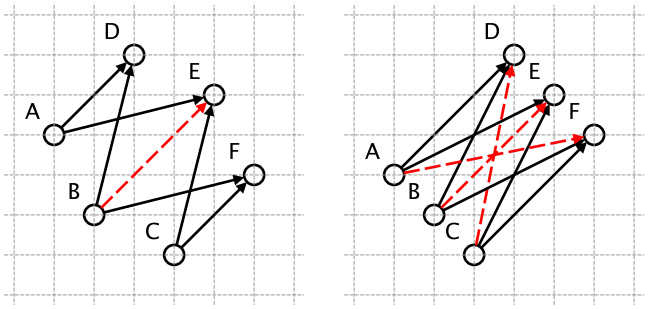

Figure 2: The "crown graph" with 6 vertices does not admit a dominance drawing. Both drawings respect the weak dominance condition. Falsely implied paths are denoted with a red dashed edge.

2. If a graph G has dimension greater than 2, then in any drawing Γ in the plane there is at least one pair of vertices $u, v \in V$ such that $X(u) \leq X(v)$ and $Y(u) \leq Y(v)$ in Γ, while neither u can reach v, nor v can reach u. Hence, we need to redefine the condition of the dominance drawing technique in order to allow all directed acyclic graphs to be visualized in the two-dimensional grid.

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. In other words, the dominance position between two vertices u and v is not supported by the existence of a real path. Such 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 [21, 22, 24], where it is shown that the corresponding decision problem is NP-complete. Heuristic approaches for minimizing the number of fips are presented in [20].

It turns out that weak dominance has many applications. For example, the fact that weak dominance can be applied to general directed acyclic graphs inspired techniques such as the one in [34] in order to answer efficiently reachability queries in very large graphs. In the work of Veloso et al. [34], X- and Y-coordinates are built according to the weak dominance condition. The benefit of such an approach is twofold; first in case the computed coordinates indicate that there is no path from u to v then the reachability query can be answered correctly with a computation that takes *constant time* (two comparisons between integers). Second, in case the computed coordinates indicate that there is a path between u and v then part of the graph can be traversed on the fly to investigate whether it is a falsely implied path. One other benefit of using the

weak dominance condition is that the search space of the traversal that looks for v can be significantly reduced in the majority of the queries since there is no point on visiting vertices that have coordinates beyond thresholds $X(v)$ and $Y(v)$ if one is looking for v. So instead of storing a complete reachability expression for a large graph which takes quadratic space, the authors use a linear amount of space and perform computations on the fly if it is needed. Combined with other heuristics that prune the search space even further, the system of [34] that utilizes the weak dominance condition outperforms the state of the art indexing algorithms with respect to the query time, the construction time, and the total size of the index in benchmarks with graphs of millions of nodes.

4 Overloaded Orthogonal Framework

The weak dominance condition lies in the core of the proposed framework for obtaining overloaded orthogonal drawings. The weak dominance condition gives us the flexibility to express all directed acyclic graphs (DAGs).

Following the footsteps of the algorithm for dominance drawing for reduced planar *st*-graphs presented in [7], 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 [7] 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 assigned to each vertex. In the third and final phase, a compaction procedure is applied to reduce the area of the drawing.

We construct a similar data structure as in "Preprocessing Step", but for general directed acyclic graphs. Let W be a rotation system of a DAG $G = (V, E)$ such that for every $u \in V$ the rotation at u contains the outgoing edges of u in consecutive order. Representation W is called a *representation in consecutive form*. The representation in consecutive form is a way to force a left-to-right order in the 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 the 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 present the algorithm for clockwise scan, that computes the X-coordinate assignment.

Algorithm 1: Topological-Sorting(W)

```
1  for each vertex v ∈ V do
2      X[v] ← ∞;
3  end
4  X[s] ← 0;
5  counter ← 1;
6  Visit-CW(s);
7  return X;
```

Algorithm 2: Visit-CW(u)

```
1  for each vertex v ∈ V such that (u, v) is the leftmost outgoing edge of u
   do
2      if in-degree(v) = 1 then
3          X[v] ← counter;
4          counter ← counter + 1;
5          remove edge (u, v);
6          Visit-CW(v);
7      else
8          remove edge (u, v);
9      end
10 end
```

Algorithm Topological-Sorting scans the outgoing edges of a vertex u in clockwise order with respect to W (leftmost outgoing edge first) 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 with respect to W, by replacing Visit-CW with Visit-CCW. The difference between Visit-CW and Visit-CCW algorithms is Line 1, where instead of leftmost outgoing edge we have rightmost outgoing edge. Note that Algorithm Topological-Sorting assigns distinct numbers to the vertices of G.

Observation: The linear ordering resulting from any topological sorting of a DAG is such that for every directed edge (u, v), vertex u comes before v in the ordering. Thus every pair of topological sortings satisfies the weak dominance condition of Section 3 using strict inequalities for the relation of the coordinates.

The topological sortings resulting from Algorithm 1 are used by WDP algorithm for assigning X- and Y-coordinates to the vertices of G.

Algorithm 3: (WDP) Weak Dominance Placement(W)

```
1  X-coordinates ← Topological-Sorting(W) using Visit-CW;
2  Y-coordinates ← Topological-Sorting(W) using Visit-CCW;
```

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 $O(n + m)$ time, algorithm WDP also runs in $O(n + m)$ time.

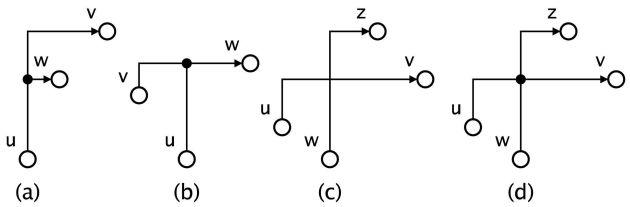

Figure 3: Examples of edges: (a) the vertical segment of edge (u, v) is overloaded (reused) by the vertical segment of edge (u, w). To visualize the edge from u to w, an e-point is placed at $(X(u), Y(w))$. (b) the horizontal segment of edge (v, w) is overloaded (reused) by the horizontal segment of (u, w). To visualize the edge from u to w, an e-point is placed at point $(X(u), Y(w))$. (c) the absence of an e-point shows that $(w, v) \notin E$, (d) the existence of an e-point in $(X(w),Y(v))$ shows that $(w, v) \in E$.

After the coordinates of the vertices have been computed, either using our WDP algorithm or any algorithm that computes the coordinates respecting the weak dominance condition, the next step is to draw the edges in a way that simplifies the drawing while clarifying information such as the existence of an edge and/or path. In order to do so, we use to our advantage the fact that each node has a unique X-coordinate and the property that all nodes that can be reached from a node u are placed in the upper-right quadrant for which u is a center. In particular all outgoing edges of a vertex use the same column in order to reach the row of their corresponding destination vertex, we illustrate this fact by using the term "overloaded" in the name of the model. We first discuss how a single edge is routed, and then we focus on unambiguously visualizing the edges of the drawing.

Edge routing is naturally 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 various segments of rows and columns we introduce e-points to resolve ambiguities, see Figure 3. We define an *e-point* as an unlabeled point that is placed on $(X(u), Y(v))$ in order to indicate a direct connection from u to v that corresponds to edge $(u, v) \in E$. Even though bends and e-points are related, we consider them to be different. Given X- and Y-coordinates there is a systematic way to describe whether a bend exists in any given point (see Lemma 1).

With this in mind we describe an algorithm that receives the vertex coordinates as input, and computes the 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 is constructed. Each vertex $u \in V$ points to the list *u.decrY* of its direct successors sorted in decreasing order according to their Y-coordinate. This single linked list *decrY* of u can be traversed by means of the operation *u.decrY.next(v)* which returns the vertex after v in the sorted list of direct suc-

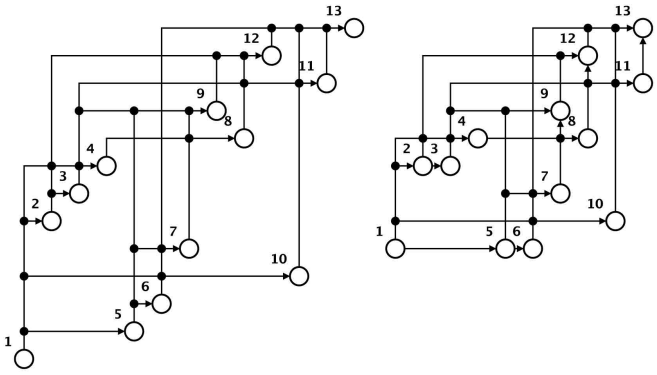

Figure 4: The overloaded orthogonal drawing of the graph of Figure 1 and its compacted version.

cessors of u. The operation *u.decrY.getFirst()* returns u's direct successor with the highest Y-coordinate (hence first in the list).

Algorithm 4: (OOD) Overloaded Orthogonal Drawing(G, $X()$, $Y()$)

```
1  for each vertex u ∈ V do
2      visited[u]← 0
3  end
4  for each vertex u ∈ V in increasing order of X-coordinate do
5      v ← u.decrY.getFirst();
6      while v ≠ null do
7          Draw edge segment from (X(u), Y(u)) to (X(u), Y(v));
8          Draw edge segment from (X(u), Y(v)) to (X(v), Y(v));
9          if u.decrY.getFirst() ≠ v ∨ visited[v] ≠ 0 then
10             New e-point ← ( X(u), Y(v) );
11         end
12         visited[v] ← 1;
13         v ← u.decrY.next(v);
14     end
15 end
```

We assume that the coordinates of each vertex are accessed in constant time (e.g., this can be achieved by using local fields for each vertex, such as $u.X$, and $u.Y$). It is clear that given a vertex placement, Algorithm 4 constructs an overloaded orthogonal drawing in linear time. In case Algorithm OOD follows the vertex placement of WDP, the obtained drawing has height $n-1$ and width $n-1$, and has at most $n-1$ bends, i.e., at most one bend for each row except for the first row.

Theorem 1 *Let $G = (V, E)$ be a directed acyclic graph where $n = |V|$ and $m = |E|$. 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.*

4.1 Transitive Closure

An interesting extension of the overloaded orthogonal drawing occurs when we include the information of the transitive closure of a directed acyclic graph in the drawing. Due to the construction of the drawing, all the reachable vertices from vertex u are located in the upper right (NE) quadrant that has the point of vertex u as a center. Therefore in case we want to extend the OOD and also depict the reachability relations among all pairs of vertices we don't have to alter the placement of the vertices of G. Specifically we can draw the additional transitive edges, that now correspond to paths, on top of an OOD, and illustrate this difference by coloring the corresponding e-point with grey color that we define as *p-point* (p stands for path). Note that due to the additional edges drawn in the T-OOD, some bends of the original OOD may become e-points in the T-OOD. This extension is called a *Transitive Overloaded Orthogonal Drawing*, or T-OOD, and it is possible due to the fact that the transitive closure of a DAG is still a DAG that preserves the reachability relations between the vertices of the original graph. In a T-OOD we can check if vertex v is reachable from vertex u by simply examining point $(X(u), Y(v))$ in the drawing. The next theorems highlight this property.

Theorem 2 *Let* Γ *be any Overloaded Orthogonal Drawing of graph* G *where the weak dominance condition holds with strict inequalities. Then there is an edge* (u, v) *in* G *if and only if there is either an e-point or a bend at the point* $(X(u), Y(v))$.

Theorem 3 *Let* Γ *be any Transitive Overloaded Orthogonal Drawing of graph* $G = (V, E)$ *where the weak dominance condition holds with strict inequalities. Then there is a path from* $u \in V$ *to* $v \in V$ *if and only if there is either a p-point (including e-points) or a bend at point* $(X(u), Y(v))$.

In case we have equalities between coordinates then the theorems still holds with the only difference that at point $(X(u), Y(v))$ there might be vertex u or vertex v.

As shown in Figure 5, the transitive edges have grey colored p-points. Notice also 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, a crossing indicates the existence of a falsely implied path. We can generalize this observation and prove the following theorem about the relation of falsely implied paths and crossings in a T-OOD of a directed acyclic st-graph.

Theorem 4 *Let* $G = (V, E)$ *be a directed acyclic st-graph. Given a vertex placement* X, Y *that respects the weak dominance condition there is a falsely implied path from* u *to* v *if and only if there exists a crossing at* $(X(u), Y(v))$ *in the Transitive Overloaded Orthogonal Drawing of* G *that is based on* X, Y.

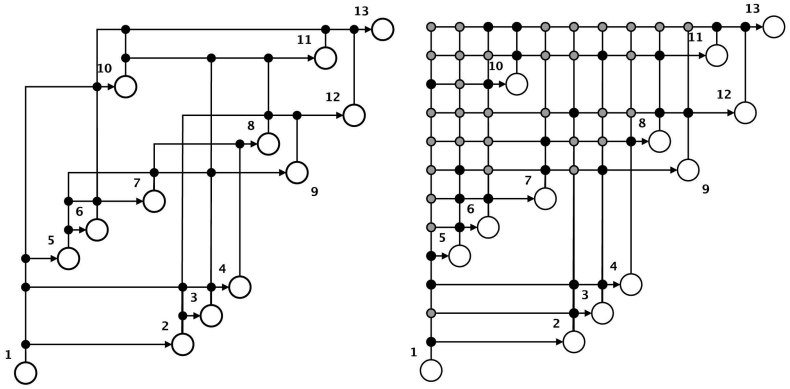

Figure 5: A transitive overloaded orthogonal drawing (right) of the graph in Figure 1. The reachability $u \rightsquigarrow v$ of any pair of vertices can be confirmed by looking at point $(X(u), Y(v))$. By the color of the point we can determine if there is an edge (e-point) or a path (p-point) between the vertices.

Proof: Assume w.l.o.g. that $X(u) \leq X(v)$ and $Y(u) \leq Y(v)$. If there is a crossing at point $(X(u), Y(v))$ then, since there is neither an e-point to denote a direct connection, nor a p-point to denote a transitive edge, we conclude that vertex u cannot reach vertex v which implies that (u, v) is a falsely implied path.

For the converse, assume there is a falsely implied path from a vertex u to a vertex v, that is, $X(u) < X(v)$ and $Y(u) < Y(v)$. Since G is a directed acyclic st-graph there must be a path (or edge) from the source s to u. Furthermore, there must be a path (or edge) from s to v which implies that in the T-OOD there exists a horizontal segment from $(X(s), Y(v))$ to $(X(v), Y(v))$ that is part of the (transitive or direct) edge (s, v). Since t is the only sink of the graph there must be a path from u to t and a path from v to t. This implies that in a T-OOD there exists a vertical segment from $(X(u), Y(u))$ to $(X(u), Y(t))$ that is a part of the (transitive or direct) edge (u, t). Therefore the horizontal segment $(X(s), Y(v))$ to $(X(v), Y(v))$ and the vertical segment $(X(u), Y(u))$ to $(X(u), Y(t))$ form a crossing at point $(X(u), Y(v))$. □

Notice that the above theorem is general in the sense that it holds *for every* vertex placement algorithm that respects the weak dominance condition. The above relation between a falsely implied path and a crossing does not necessarily hold for every vertex placement when we are dealing with directed acyclic graphs with multiple sources and multiple sinks. An easy counterexample can be constructed in the following way: Let $G = (V, E)$ be a DAG with a set of sources $S = \{s_1, \ldots, s_k\}$. Let also X, Y be a vertex placement such that $X(s_i) < X(s_j)$ and $Y(s_i) < Y(s_j)$, where $i, j \in \{1, ..., k\}$. Notice that no vertex with X-coordinate less than $X(s_j)$ has an outgoing edge to source s_j. Thus, there is no horizontal edge segment on row $Y(s_j)$ that extends to column $X(s_j)$. Therefore there exists a falsely implied path from s_i to s_j but there is no crossing

on point $(X(s_i), Y(s_j))$ of the corresponding T-OOD.

In general, a falsely implied path (u, v) does not correspond to a crossing in a T-OOD if any of the following holds:

- There is no path from vertex u to any of the vertices that have Y-coordinate greater than $Y(v)$.
- There is no path from vertices that have X-coordinate less than $X(u)$ to v.

5 Properties of the Drawing

In this section we discuss properties and bounds of overloaded orthogonal drawings for directed acyclic graphs. Certain conditions must hold in order to form a bend.

Lemma 1 *Let $G = (V, E)$ be a directed acyclic graph and $(u, v) \in E$ where $u, v \in V$. Edge (u, v) yields a bend in any OOD Γ of G with coordinate assignment (X, Y), if and only if all the following three conditions hold:*

- *u is the predecessor of v with the minimum X-coordinate*
- *v is the successor of u with the maximum Y-coordinate*
- *$X(u) \neq X(v)$ and $Y(u) \neq Y(v)$*

Proof: For the 'if' part, let us assume that the above three conditions hold. Then according to the third condition, u cannot be on the same vertical line (i.e., column) as v due to $X(u) \neq X(v)$. Also, u cannot be on the same horizontal line (i.e., row) as v due to $Y(u) \neq Y(v)$. Thus, point $(X(u), Y(v))$ contains either a bend or an e-point. If there was an e-point on $(X(u), Y(v))$ there would be either a predecessor of v namely w such that $X(w) < X(u)$, or a successor of u namely z such that $Y(v) < Y(z)$, or both. But that contradicts the first two conditions. Thus, edge (u, v) yields a bend.

For the 'only if' part, let's assume that edge (u, v) yields a bend, we prove that the above three conditions hold. Vertex u is the direct predecessor of v with the minimum X-coordinate, or else there would be an e-point instead of a bend in $(X(u), Y(v))$. Similarly, vertex v is the direct successor of u with the maximum Y-coordinate, or else there would be an e-point instead of a bend in $(X(u), Y(v))$. Finally, since there is a bend we have $X(u) \neq X(v)$ and $Y(u) \neq Y(v)$. Therefore, the above three conditions hold. □

Thus, if $X(u) \neq X(v)$ and $Y(u) \neq Y(v)$ for every pair of vertices $u, v \in V$, we conclude that every edge has a "step-like" form and consequently produces either a bend or an e-point. Therefore we have the following:

Lemma 2 *Let Γ be an overloaded orthogonal drawing of a directed acyclic graph G with m edges, where each vertex is placed in distinct X-, Y- coordinates. Then the following holds: $bends(\Gamma) + ePoints(\Gamma) = m$.*

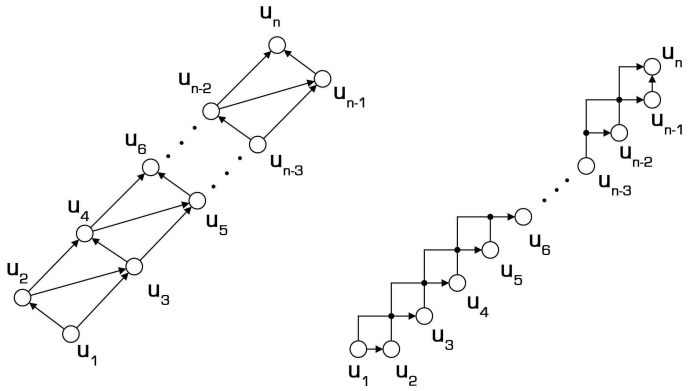

Figure 6: An illustration for the construction in the proof of Theorem 5.

Every vertex $v \in G$ can have at most one bend on its row $Y(v)$. That bend is produced from the direct predecessor of v with the lowest X-coordinate. Taking into consideration that sources do not have incoming edges, we immediately have the following lemma:

Lemma 3 *Let Γ be any overloaded orthogonal drawing of a DAG G with n vertices and m edges. Let also n_s be the number of sources of G. Then $bends(\Gamma) \leq n - n_s$.*

Proof: Among the incoming edges of a vertex only one can have a bend in Γ. Taking into consideration that sources do not have incoming edges, every overloaded orthogonal drawing has at most $n - n_s$ bends. □

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

Theorem 5 *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: Consider the graph G_n shown in Figure 6. Each vertex u_i has two outgoing edges (u_i, u_{i+2}) and (u_i, u_{i+1}), where $i < n-1$. The transitive closure of each graph in this family is a DAG in which there is a directed edge for every unordered pair of vertices, therefore there is only one possible topological sorting and is used for both X- and Y-coordinates. According to our assumption there are no empty rows and columns, thus, there is only one possible OOD Γ with distinct X-, Y-coordinates for the graph G_n. If one considers the possibility of compacting the drawing, then the drawing admits 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. The allowable compactions for any OOD, as well as a proposed algorithm, are discussed in detail in Section 6.

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. □

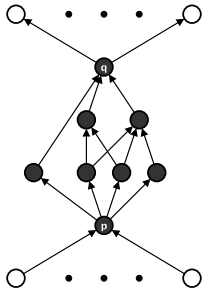

Figure 7: An illustration of a pq-component of a single-source single-sink DAG. Every path from the source to a grey node passes through p, and every path from a grey node to the sink passes through q.

In the rest of this section we discuss a property of the suggested vertex placement algorithm WDP. Specifically, the vertex placement by Algorithm WDP creates a natural separation between pq-components in single-source single-sink DAGs. A *pq-component* $G_{pq} = (V', E')$ is an induced subgraph of a single-source single-sink DAG G for which the following properties hold: 1) it contains at least two edges, 2) vertex $p \in V'$ is a dominator of every vertex $v \in V'$ and $q \in V'$ is a post-dominator of every vertex $v \in V'$, 3) for every outgoing edge $(p, v) \in E$ of p, we have $v \in V'$ and 4) for every incoming edge $(u, q) \in E$ of q, we have $u \in V'$. An illustration of a pq-component is presented in Figure 7.

The term dominator, correspondingly post-dominator, is capturing a completely different notion of dominance compared to the dominance drawings (see also Section 2). The dominator of a node u is an intermediate node that participates in all the paths that connect the source of the graph with u. On the other hand when v dominates u in a dominance drawing, the following relation holds for their coordinates $X(u) \leq X(v)$ and $Y(u) \leq Y(v)$. In this case the dominance notion refers to their coordinate comparison.

Lemma 4 *Let $G = (V, E)$ be a directed acyclic graph with a single-source and a single-sink that includes a pq-component $G_{pq} = (V', E')$. Let X, Y be the vertex placement resulting from WDP. Then for every $w \in V'$ the following inequalities hold: $X(p) \leq X(w) \leq X(p) + |V'| - 1$ and $Y(p) \leq Y(w) \leq Y(p) + |V'| - 1$.*

Proof: Vertex q is reachable from p, because q is reachable from all the vertices in V' and vertex p can reach all the vertices in V'. From the fact that vertex p is a dominator of every vertex in V' we can derive that there are no edges (u, v) such that $u \in V - V'$ and $v \in V' - \{p\}$. This implies that for every $w \in V' - \{p\}$ the incoming edges of w have a vertex in V' as an origin. Furthermore, considering

the fact that vertex q is a post-dominator of every vertex in V' we can derive that there are no edges (u, v) such that $u \in V' - \{q\}$ and $v \in V - V'$. Thus, for all $u \in V' - \{q\}$ the outgoing edges of u have a vertex in V' as a destination. Using the above arguments we can see that during the iteration in which the algorithm WDP visits vertex p, all $|V'|-1$ vertices of G' are visited consecutively within the next $|V'|-1$ calls of method Visit-CW. Thus, for every vertex $w \in V'$, $X(p) \leq X(w) \leq X(p) + |V'| - 1$. The same line of reasoning can be followed to prove that all $|V'| - 1$ vertices of G' are visited within the next $|V'| - 1$ calls of method Visit-CCW from the time the algorithm WDP visits p. It follows that for every $w \in V'$, $Y(p) \leq Y(w) \leq Y(p) + |V'| - 1$. □

Let $X()$ and $Y()$ be coordinates that respect the weak dominance condition. Also let $G' = (V', E')$ be a component where $V' \subseteq V$ and $E' \subseteq E$. A component G' is said to be *separated with respect to X() and Y()*, 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*. We focus on pq-components and their properties but depending on the application area one can propose different component definitions and placement algorithms that highlight their structural properties.

Theorem 6 *The vertex placement $X()$ and $Y()$ constructed by algorithm Weak Dominance Placement respects the separation property for every pq-component.*

Proof: Let $G' = (V', E') \subseteq G$ be a pq-component. According to Lemma 4 algorithm WDP for G returns a numbering of vertices of G' from $X(p)$ to $X(p) + |V'| - 1$. This also holds for Y-coordinates, i.e., numbers vertices of G' from $Y(p)$ to $Y(p) + |V'| - 1$. Thus, for every vertex u such that $u \in V - V'$ we have $X(u) \notin [X(p), X(q)]$ and $Y(u) \notin [Y(p), Y(q)]$. □

Let $G_{pq} = (V', E')$ be a pq-component, in case V' contains another pq-component(s) algorithm WDP respects the separation property for the internal pq-component(s). In other words the separation property holds recursively for all sub-components.

6 Compaction

Our approach is to apply a post-processing compaction step to an overloaded orthogonal drawing. A different approach would be to design a vertex placement algorithm that respects the weak dominance condition while optimizing the area, as opposed to compacting as a post-processing step (which is done here). This approach is not explored in this work.

In this section we discuss how to reduce the area of a drawing by allowing vertices to share the same X- or Y-coordinate, i.e., allowing equality between

the coordinates of two vertices. The proposed compaction algorithm can be deployed as an independent post-processing step that is not related to the vertex placement algorithm that was used as long as the coordinate assignment satisfies the weak dominance condition. Without loss of generality we assume that there are no empty rows and columns, if there are any we simply remove them. Our compaction follows the steps of the compaction algorithm in [6, 7]. However since our graphs are not planar, and therefore we do not have planar embeddings, we need to make some adjustments in order to produce a valid drawing.

We proceed by describing the intuition of the compaction algorithm. Given an overloaded orthogonal drawing our algorithm visits the vertices in increasing order of X-coordinate. Let u be the vertex that is currently visited, then the algorithm checks whether vertex v with the next highest X-coordinate can be "pushed" to the left in order for u and v to share the same column $X(u)$. If the X-coordinate of v is changed to $X(u)$, all the e-points and bends of column $X(v)$ should be pushed to the left too. Notice that if we stored the exact coordinates of an e-point/bend we would need constant time for the move of each e-point/bend. Instead, if we store a reference to the corresponding node of the column for each e-point/bend, then we can move all the e-points/bends of a column in constant time by just changing the coordinates of the node in that column. A similar procedure is followed for the Y-coordinate. Roughly speaking, the compaction step "straightens" a set of 1-bend polyline edges. An example is shown in Figure 4.

Compaction Invariant Conditions: More formally, let Γ be an OOD of a directed acyclic graph G then for the X- and Y-coordinate assignment the following conditions hold:
1) Vertices u, v have the same X-coordinate (resp. Y-coordinate) only if $(u, v) \in E$.
2) No vertices u, v coincide at the same point of the grid.
3) No vertex has coordinates that are within the range of a horizontal/vertical segment of an edge. That is, there is no $(u, v) \in E$ and $w \in V - \{u, v\}$ such that $((X(u) = X(w)) \wedge (Y(u) < Y(w) \leq Y(v))$ or $(X(u) \leq X(w) < X(v)) \wedge (Y(w) = Y(v))$.

The last condition prohibits the scenarios where a compaction step forces an edge to *"pass over"* a vertex. It is easy to see that if the weak dominance condition holds with strict inequalities, i.e., distinct X-, Y-coordinates, then the compaction invariant conditions hold. Also notice that if compaction is performed on Γ with respect to the compaction invariant conditions, then the sum $bends(\Gamma) + ePoints(\Gamma)$ would be less than the number of edges.

We note here that the coordinate assignment that is given as an input to X-compaction (resp. Y-compaction) algorithm should have *distinct* X-coordinates (resp. Y-coordinates). Let incListX be a linked list of vertices such that vertices are arranged in the list in *increasing* order of X-coordinate. The linked list incListY is defined in a similar manner.

Algorithm 5: X-compaction(G, $X()$, $Y()$)

```
1  u ← incListX.firstNode();
2  while incListX.next(u) ≠ null do
3      v ← incListX.next(u);
4      if successorHighestY(u)=v ∧ (u, v) ∈ E ∧ Y(v) ≠ Y(u) then
5          if indegree(v) > 1 then
6              remove the e-point in (X(u), Y(v));
7          else
8              remove the bend in (X(u), Y(v));
9          end
10         X(v) ← X(u);
11         Move e-points and bends of column X(v) to column X(u);
12     else
13         X(v) ← X(u) + 1;
14         Move e-points and bends of column X(v) to column X(u) + 1;
15     end
16     u ←incListX.next(u);
17 end
```

Lemma 5 *Let $X(), Y()$ be the coordinate assignment of an overloaded orthogonal drawing of a DAG G such that (a) every vertex has distinct X-coordinates and (b) $X()$ and $Y()$ respect the coordinate invariant conditions. Then Algorithm X-compaction$(G, X(), Y())$ returns a coordinate assignment that respects the compaction invariant conditions.*

Proof: It is straight-forward to show that conditions 1) and 2) hold for the output of X-compaction. Therefore we prove that if a compaction in X-coordinate is performed between vertices u, v then no vertex has coordinates that are within the range of a horizontal/vertical segment of an edge (we refer to this event as "edge passing over a vertex"). According to our assumption in the beginning of the section (w.l.o.g.) there are no empty rows and columns. Suppose for the sake of contradiction that the first event of an edge passing over a vertex takes place after the compaction in X-coordinate between u, v, which we call compaction $u - v$ for brevity. We call $e' \in E$ the edge that passes over a vertex, call it $z \in V$. Notice that the steps in Lines 10, 11 during compaction $u - v$ do not move the vertices that have X-coordinate smaller than $X(u)$ and the vertices that have X-coordinate greater than $X(v)$. Thus z should be either in column $X(u)$ (i.e., $X(z) = X(u)$) or column $X(v)$ (i.e., $X(z) = X(v)$).

Case $X(z) = X(u)$: If vertex z were already in column $X(u)$ before compaction $u - v$ it must be the case that by moving e' to column $X(u)$ we forced e' to pass over z. It is not hard to show that if there are more than one vertices in column $X(u)$, due to previous compaction steps of the algorithm, then u is the vertex with the highest Y-coordinate. From the above observation $Y(z)$ should be smaller than $Y(u)$ (see Figure 8 (a)). But then in order for e' to pass over z the origin of e' should be in column $X(v)$. This contradicts the assumption that the X-coordinates of the input are distinct.

Case $X(z) = X(v)$: If vertex z were in column $X(v)$ before compaction

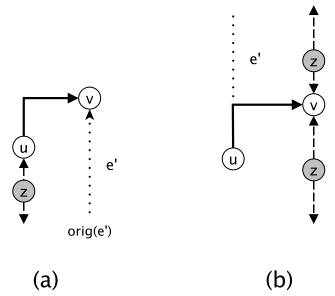

Figure 8: Illustration of the Proof of Lemma 5. Figure (a) illustrates the case where $X(z) = X(u)$ and Figure (b) the case where $X(z) = X(v)$.

$u - v$ it must be the case that the vertical segment of e' was already in column $X(u)$ before compaction $u - v$. Thus by moving z to column $X(u)$ we forced the vertical segment of e' to pass over z. We continue with a case analysis on the Y-coordinate of z. The case where $Y(z)$ is less than $Y(v)$ (see Figure 8 (b)) implies that z and v are in the same column since u and v are consecutive on the X-axis. This contradicts the assumption that the X-coordinates of the input are distinct. So we focus on the case where $Y(z)$ is greater than $Y(v)$. In order for e' to pass over z it must be the case that $orig(e') = u$. But if u can reach a vertex that has Y-coordinate greater than v we contradict the condition of Line 4 that requires successorHighestY(u)=v.

Both of the above cases reach a contradiction which shows that there is no e' that passes over a vertex z in the output of X-compaction. □

Using a symmetric argument one can prove that the output of Algorithm Y-compaction respects the compaction invariant conditions.

Algorithm 6: Y-compaction(G, $X()$, $Y()$)

```
1  u ← incListY.firstNode();
2  while incListY.next(u) ≠ null do
3      v ← incListY.next(u);
4      if predecessorLowestX(v) = u ∧ (u, v) ∈ E ∧ X(v) ≠ X(u) then
5          if outdegree(u) > 1 then
6              remove the e-point in (X(u), Y(v));
7          else
8              remove the bend in (X(u), Y(v));
9          end
10         Y(v) ← Y(u);
11         Move all e-points and bends from row Y(v) to row Y(u);
12     else
13         Y(v) ← Y(u) + 1;
14         Move all e-points and bends from row Y(v) to row Y(u) + 1;
15     end
16     u ←incListY.next(u);
17 end
```

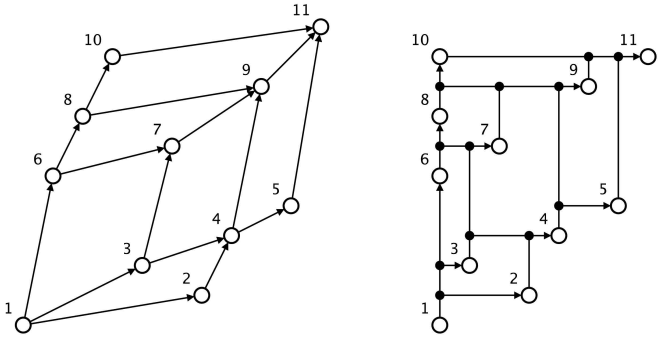

Figure 9: In the left figure we have the straight-line dominance drawing of a reduced planar *st*-graph as described in [6]. In the right figure there is a compacted overloaded orthogonal drawing of the same graph with zero bends.

Lemma 6 *Let $X(), Y()$ be the coordinate assignment of an overloaded orthogonal drawing of a DAG G such that (a) every vertex has distinct Y-coordinates and (b) $X()$ and $Y()$ respect the coordinate invariant conditions. Then Algorithm Y-compaction$(G, X(), Y())$ returns a coordinate assignment that respects the compaction invariant conditions.*

By combining the regular dominance drawing technique (vertex placement) applied to reduced planar *st*-graphs as presented in [7], and the overloaded edge routing technique of this work we obtain an overloaded orthogonal drawing with zero bends, an example is depicted in Figure 9.

Theorem 7 *Given a reduced planar st-graph $G = (V, E)$ with n vertices, an overloaded orthogonal drawing Γ of G with zero bends can be constructed in $O(n)$ time.*

Proof: Consider a reduced planar *st*-graph G. Assume that the coordinates of the vertices are obtained by the dominance drawing algorithm in [7] which implies that the drawing Γ has no falsely implied paths. Then a vertex v is reachable from u if and only if $X(u) \leq X(v)$ and $Y(u) \leq Y(v)$. For the sake of contradiction we assume that there is an edge $(u, w) \in E$ such that it has a bend that cannot be removed during the compaction phase. In order to have a bend all three conditions of Lemma 1 must hold. Recall that G is embedded in the plane with s and t on the external face. We denote as $u \to v$ the existence of a path from u to v. By following the above assumptions, we will conclude that there cannot be any edge (u, w) that forms a bend which we cannot remove.

Let us assume that all conditions of Lemma 1 hold for edge (u, w). According to the third condition $X(u) \neq X(w)$ and $Y(u) \neq Y(w)$. If vertices u, w were consecutive in X-coordinate or Y-coordinate, then we could eliminate the bend performing a compaction step on X, Y respectively. But this contradicts the

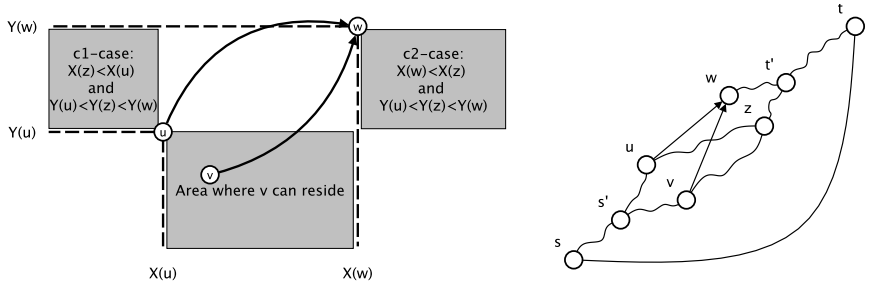

Figure 10: Proof of Theorem 7. The left figure illustrates the assumptions about the relation of the involved vertices prior to the analysis of $c1$-case and $c2$-case. The right figure shows a drawing of a $K_{3,3}$ that results from $c2$-case.

assumption that the bend cannot be removed during the compaction phase. Thus, vertices u, w cannot be consecutive neither in X- nor in Y-coordinate. We are going to use this fact in the rest of the proof. Notice also that the set of direct predecessors of w must contain more than one vertex, otherwise the vertex placement algorithm would have assigned consecutive coordinates to vertices u, w.

Let $v \in V$ be a direct predecessor of w such that $v \neq u$, we reason about the properties that hold for the coordinates of v. Taking into consideration that the first condition of Lemma 1 holds for the bend (u, w), the inequality $X(u) < X(v) \leq X(w)$ must hold. As for the Y-coordinate, $Y(v)$ cannot be in the range $[Y(u), Y(w)]$ because in that case edge (u, w) would be transitive, contradiction. Therefore we have $Y(v) \leq Y(u)$. We have shown that vertices u, w are not consecutive in X-coordinate, but we haven't argued yet about the relation of the two vertices in the Y-coordinate. Since they can't be consecutive (otherwise we would remove the bend), there must be a vertex z such that $Y(u) < Y(z) \leq Y(w)$. Then vertex z cannot be in the range $X(u) < X(z) < X(w)$, because in that case the edge (u, w) is a transitive edge, contradiction. Thus, we have two cases to consider, let $c1$ be the case where $X(z) < X(u)$ and $c2$ the case where $X(z) > X(w)$. The left picture of Figure 10 illustrates the assumptions made up to this point about the relation of the involved vertices.

Case c1 ($X(z) < X(u)$): In this case we have $X(z) < X(w)$ and $Y(z) < Y(w)$, therefore vertex w is reachable from vertex z, $z \to w$. But, in order to follow the first condition of Lemma 1, we have already assumed that vertex u is the direct predecessor of w with the minimum X-coordinate. Hence, z can not be a direct predecessor of w since this would contradict the assumption that u is the direct predecessor of w with the minimum X-coordinate. Also z cannot reach u due to the fact that $Y(u) < Y(z)$ and that would violate the weak dominance condition. Since we showed that $z \to w$ but z can not be a direct predecessor of w, there must be another direct predecessor of w namely q that is reachable from z, that is $z \to q$ and $(q, w) \in E$. As for the coordinates of q,

$X(z) < X(u) < X(q) \leq X(w)$ and $Y(u) < Y(z) < Y(q) < Y(w)$. Taking into consideration the fact that we have zero falsely implied paths and $X(u) < X(q)$ and $Y(u) < Y(q)$, vertex q must also be reachable from u, $u \rightarrow q$. But q is a direct predecessor of w, so from the fact that $u \rightarrow q$ and the existence of edge (q, w) we can conclude that edge (u, w) is transitive, contradiction.

Case c2 ($X(z) > X(w)$): In this case vertex z is reachable from u, since $X(z) > X(w) > X(v) > X(u)$ and $Y(w) > Y(z) > Y(u) > Y(v)$. From the domination of the coordinates, vertex z is also reachable from v. Consider paths $s \rightarrow u$ and $s \rightarrow v$, and let s' be the last (farthest from s) vertex common to both paths. Likewise, let t' be the first (farthest from t) vertex common to paths $w \rightarrow t$ and $z \rightarrow t$. By the above definitions and the dominance property, G has the following paths:

$$s' \rightarrow u \ , \ s' \rightarrow v \ , \ w \rightarrow t' \ , \ z \rightarrow t' \ ,$$

$$u \rightarrow w \ , \ u \rightarrow z \ , \ v \rightarrow w \ , \ v \rightarrow z \ .$$

Since s and t are on the external face, we can add the edge (s, t) to G, while preserving planarity of the embedding. One can easily verify that the paths listed above, plus edge (s, t) form a graph that is homomorphic to $K_{3,3}$. This fact contradicts the planarity of G. □

7 Other Graphs

In this section we discuss methods that extend the proposed overloaded orthogonal model in order to handle undirected graphs and directed graphs with cycles. A visualization framework based on the following extensions was originally proposed in [23].

7.1 Undirected Graphs

Let G be an undirected graph and s, t be two distinct vertices of G. 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 [11]. 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 [31, 32], parametrically controls the length of the longest path of the final st-oriented graph. Specifically, the parameter $0 \leq p \leq 1$ is given as an input to the algorithm in order to control the length of the longest path. For $p = 1$ the algorithm produces st-oriented graphs of long longest path whereas for $p = 0$ the algorithm produces st-oriented graphs of small longest path. 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. For the undirected graph of Figure 11 the st-orientation with $p = 0$ results in a compacted

overloaded orthogonal drawing with area 11×11, while the *st*-orientation with $p = 1$ results in a compacted overloaded orthogonal drawing with optimal area 2×11.

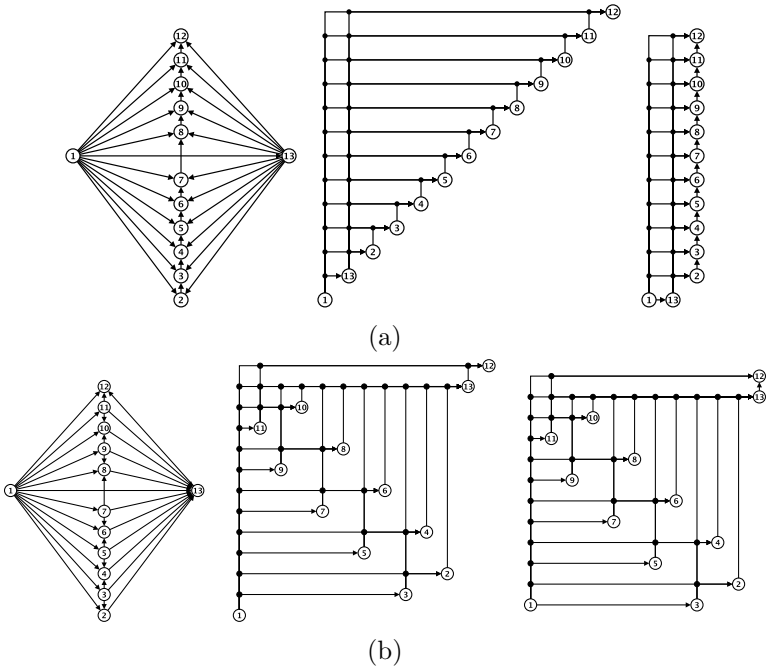

Figure 11: Overloaded orthogonal drawings of an originally undirected planar graph are shown. In (a) the *st*-orientation was produced by algorithm [31] with parameter $p = 1$. In (b) the *st*-orientation was produced with parameter $p = 0$. As one can see from the third drawing of (a) & (b) the value of p affects the compacted OOD.

7.2 Directed Graphs with Cycles

A cycle is clearly not representable in a drawing that respects the weak dominance property. Therefore, we follow a well known technique that temporarily inverts the direction of a set of edges such that the graph becomes a directed acyclic graph. This set of edges is also known as a feedback arc set and as described in Section 2, minimizing the feedback arc set is NP-hard. In the rest of the section we deploy a heuristic algorithm [6] that computes a minimal feedback arc set. The better the approximation algorithm for the minimization of the feedback arc set, the smaller the number of inverted edges (see description later).

First we compute the minimal feedback arc set and denote it as E'. Next, the direction of the edges in E' is inverted (or the edges in E' can be even

removed if there are no connectivity issues). The OOD algorithm can be used now to visualize the resulting directed acyclic graph since it contains no cycles. We denote as Γ the resulting drawing of the DAG. We focus on how to visualize an edge $(v, u) \in E'$ that belongs to the feedback arc set. Notice that even though u is reachable from v, node u is placed in the lower-left quadrant of v.

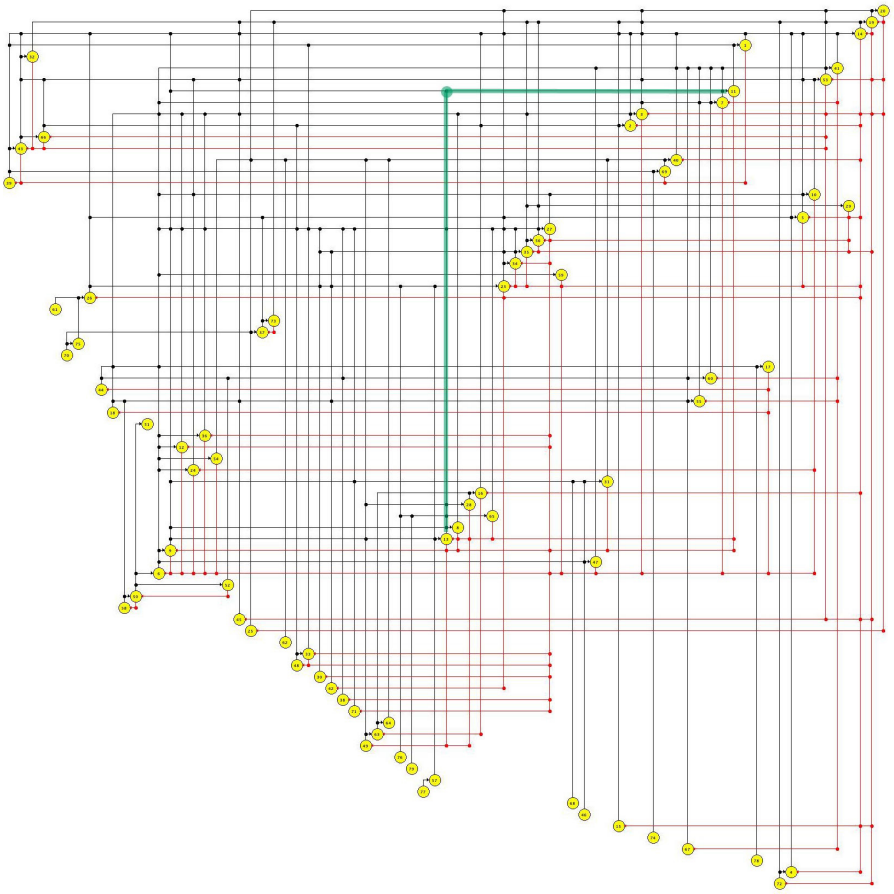

Figure 12: An overloaded orthogonal drawing of a directed graph with 79 nodes and 335 edges. The feedback arc set is colored red.

Also note that all the incoming edges of node u in Γ are assembled in the row $Y(u)$ of the grid. Thus, the $Y(u)$ row has no edge segment to the right of point $(X(u), Y(u))$. Similarly, all the outgoing edges of node v are assembled in the column $X(v)$, above the point of node v of the grid. Thus, column $X(v)$ has no edge segment below point $(X(v), Y(v))$. We use these "empty" segments (1) to the right of u and (2) below v, to draw the feedback arc (v, u). But with this edge visualization of (v, u) we violate the weak dominance property. In order

to highlight this violation, the edges in E' are colored red and are oriented from top-to-bottom and from right-to-left. Formally, each edge $e' = (v, u) \in E'$ has a vertical segment from point $(X(v), Y(v))$ to point $(X(v), Y(u))$, and a horizontal segment from point $(X(v), Y(u))$ to point $(X(u), Y(u))$. Given a feedback arc (v, u) the corresponding e-point is defined as an unlabeled *red point* that is placed on point $(X(v), Y(u))$ to indicate a direct connection from v to u. Figure 12 shows an illustration of an overloaded orthogonal drawing of a directed graph with cycles.

8 Conclusions: Clarity and Readability of the Model and Open Problems

We presented algorithms that produce overloaded orthogonal drawings of any directed acyclic graph $G = (V, E)$ with at most $n - 1$ bends, $O(n^2)$ area, where $n = |V|$ and $m = |E|$. Our algorithms run in linear $O(n + m)$ time, and are easy to implement. In particular, we outline some advantages of the overloaded orthogonal drawing model.

• *Meaningful relation between vertex coordinates*: The weak dominance condition implies that if there is a path from u to v then vertex v appears 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 choose any pair of topological sortings as X, Y coordinates.

• *Universality of the model*: The overloaded orthogonal model handles the same way 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 by computing an *st*-numbering as a pre-processing step. Different *st*-numberings result in different OODs. This allows the visualization of interesting characteristics of the graph (i.e., longest paths [31, 32]).

• *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 have to visually follow every outgoing edge of u successively, until we find the one that reaches 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*.

• *Efficient Visual Confirmation of Reachability*: An interesting extension of this graph drawing technique occurs by including the information of the transitive closure of a graph. In that case every possible path along the original directed acyclic graph $G = (V, E)$ is represented by an e-point that can be colored with either black (if it corresponds to an edge) or grey (if it corresponds to a path). 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. In this context, crossings indicate the existence of falsely implied paths.

There are several interesting open problems remaining: (1) propose better heuristics and approximation algorithms for reducing the number of falsely implied paths; (2) propose algorithms for weak dominance placement that provide upper bounds on the number of crossings in an overloaded orthogonal drawing of the transitive closure; (3) use the weak dominance condition in order to construct enhanced data structures than can improve reachability queries in very large graphs where the computation and storage of the complete transitive closure might be practically impossible.

Acknowledgements

We would like to thank the two anonymous reviewers whose comments and suggestions improved our paper.

References

[1] W. Barth, P. Mutzel, and C. Yildiz. A new approximation algorithm for bend minimization in the Kandinsky model. In *Proc. 14th International Symposium on Graph Drawing (GD'06)*, pages 343–354. LNCS 4372, Springer-Verlag, 2006. doi:10.1007/978-3-540-70904-6_33.

[2] 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. doi:10.1109/12.868028.

[3] T. Biedl and G. Kant. A better heuristic for orthogonal graph drawings. *Computational Geometry: Theory and Applications*, 9(3):159–180, 1998. doi:10.1016/S0925-7721(97)00026-6.

[4] T. C. Biedl, B. P. Madden, and I. G. Tollis. The three-phase method: A unified approach to orthogonal graph drawing. *International Journal of Computational Geometry and Applications*, 10(6):553–580, 2000. doi:10.1142/S0218195900000310.

[5] T. Bläsius, G. Brückner, and I. Rutter. Complexity of higher-degree orthogonal graph embedding in the Kandinsky model. In *Proc. 22nd European Symposium on Algorithms (ESA'14)*, pages 161–172. LNCS 8737, Springer Berlin Heidelberg, 2014. doi:10.1007/978-3-662-44777-2_14.

[6] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing: Algorithms for the Visualization of graphs*. Prentice - Hall, New Jersey, U.S.A., 1998.

[7] G. Di Battista, R. Tamassia, and I. G. Tollis. Area requirement and symmetry display of planar upward drawings. *Discrete and Computational Geometry*, 7(4):381–401, 1992. doi:10.1007/BF02187850.

[8] M. Dickerson, D. Eppstein, M. T. Goodrich, and J. Y. Meng. Confluent drawings: Visualizing non-planar diagrams in a planar way. *Journal of Graph Algorithms and Applications*, 9(1):31–52, 2005. doi:10.7155/jgaa.00099.

[9] P. Eades, H. ElGindy, M. Houle, B. L. M. Miller, D. Rappaport, and S. Whitesides. Dominance drawings of bipartite graphs. Technical report, 1994.

[10] D. Eppstein, M. T. Goodrich, and J. Y. Meng. Confluent layered drawings. *Algorithmica*, 47(4):439–452, 2007. doi:10.1007/s00453-006-0159-8.

[11] S. Even and R. E. Tarjan. Computing an *st*-numbering. *Theoretical Computer Science*, 2(3):339–344, 1976. doi:10.1016/0304-3975(76)90086-4.

[12] U. Fößmeier and M. Kaufmann. Drawing high degree graphs with low bend numbers. In *Proc. 4th International Symposium on Graph Drawing (GD'96)*, pages 254–266. LNCS 1190, Springer, 1996. doi:10.1007/BFb0021809.

[13] U. Fößmeier and M. Kaufmann. Algorithms and area bounds for nonplanar orthogonal drawings. In *Proc. 5th International Symposium on Graph Drawing (GD'97)*, pages 134–145. LNCS 1353, Springer-Verlag, 1997. doi:10.1007/3-540-63938-1_57.

[14] E. R. Gansner, Y. Hu, S. North, and C. Scheidegger. Multilevel agglomerative edge bundling for visualizing large graphs. In *Proc. of the 2011 IEEE Pacific Visualization Symposium (PACIFICVIS '11)*, pages 187–194. IEEE Computer Society, 2011. doi:10.1109/PACIFICVIS.2011.5742389.

[15] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, U.S.A., 1979.

[16] L. Georgiadis, R. E. Tarjan, and R. F. Werneck. Finding dominators in practice. *Journal of Graph Algorithms and Applications*, 10(1):69–94, 2006. doi:10.7155/jgaa.00119.

[17] D. Holten. Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):741–748, 2006. doi:10.1109/TVCG.2006.147.

[18] D. Holten and J. J. Van Wijk. Force-directed edge bundling for graph visualization. *Computer Graphics Forum (Proc. EuroVis)*, 28(3):983–990, 2009. doi:10.1111/j.1467-8659.2009.01450.x.

[19] R. M. Karp. Reducibility among combinatorial problems. In *Proc. of a Symposium on the Complexity of Computer Computations*, pages 85–103. 1972. doi:10.1007/978-1-4684-2001-2_9.

[20] E. M. Kornaropoulos. Dominance drawing of non-planar graphs. M.Sc. Thesis, Computer Science Department, University of Crete, Greece, 2012.

[21] E. M. Kornaropoulos and I. G. Tollis. Overloaded orthogonal drawings. In *Proc. 19th International Symposium on Graph Drawing (GD'11)*, pages 242–253. LNCS 7034, Springer-Verlag, 2011. doi:10.1007/978-3-642-25878-7_24.

[22] E. M. Kornaropoulos and I. G. Tollis. Weak dominance drawings and linear extension diameter. arXiv:1108.1439, 2011.

[23] E. M. Kornaropoulos and I. G. Tollis. Dagview: An approach for visualizing large graphs. In *Proc. 20th International Symposium on Graph Drawing (GD'12)*, pages 499–510. LNCS 7704, Springer-Verlag, 2012. doi:10.1007/978-3-642-36763-2_44.

[24] E. M. Kornaropoulos and I. G. Tollis. Weak dominance drawings for directed acyclic graphs. In *Proc. 20th International Symposium on Graph Drawing (GD'12)*, pages 559–560. LNCS 7704, Springer-Verlag, 2012. doi:10.1007/978-3-642-36763-2_52.

[25] A. Lambert, R. Bourqui, and D. Auber. Winding roads: routing edges into bundles. In *Computer Graphics Forum*, pages 853–862. 29(3), 2010. doi:10.1111/j.1467-8659.2009.01700.x.

[26] T. Lengauer. *Combinatorial algorithms for integrated circuit layout*. John Wiley & Sons, Inc., New York, NY, 1990.

[27] B. Mohar and C. Thomassen. *Graphs on Surfaces*. Johns Hopkins Series in the Mathematical Sciences, Johns Hopkins University Press, 2001.

[28] K. Nomura, S. Tayu, and S. Ueno. On the orthogonal drawing of outerplanar graphs. *Journal IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E88-A(6):1583–1588, 2005. doi:10.1093/ietfec/e88-a.6.1583.

[29] A. Papakostas and I. G. Tollis. Algorithms for area-efficient orthogonal drawings. *Computational Geometry Theory and Applications*, 9(1-2):83–110, 1998. doi:10.1016/S0925-7721(97)00017-5.

[30] A. Papakostas and I. G. Tollis. Efficient orthogonal drawings of high degree graphs. *Algorithmica*, 26(1):100–125, 2000. doi:10.1007/s004539910006.

[31] C. Papamanthou and I. G. Tollis. Algorithms for computing a parameterized *st*-orientation. *Theoretical Computer Science*, 408(2-3):224–240, 2008. doi:10.1016/j.tcs.2008.08.012.

[32] C. Papamanthou and I. G. Tollis. Applications of parameterized *st*-orientations. *Journal of Graph Algorithms and Applications*, 14(2):337–365, 2010. doi:10.7155/jgaa.00210.

[33] M. S. Rahman, T. Nishizeki, and M. Naznin. Orthogonal drawings of plane graphs without bends. *Journal of Graph Algorithms and Applications*, 7(4):335–362, 2003. doi:10.7155/jgaa.00074.

[34] R. Rodrigues Veloso, L. Cerf, W. Meira, Jr., and M. J. Zaki. Reachability queries in very large graphs: A fast refined online search approach. In *Proc. 17th International Conference on Extending Database Technologies (EDBT'14)*, pages 511–522. 2014. doi:10.5441/002/edbt.2014.46.

[35] D. Selassie, B. Heller, and J. Heer. Divided edge bundling for directional network data. *IEEE Transactions on Visualization & Computer Graphics*, vol.17, no.12:2354–2363, 2011. doi:10.1109/TVCG.2011.190.

[36] J. Storer. On minimal node-cost planar embeddings. *Networks*, 14(2):181–212, 1984. doi:10.1002/net.3230140202.

[37] R. Tamassia. On embedding a graph in the grid with the minimum number of bends. *SIAM Journal on Computing*, 16(3):421–444, 1987. doi:10.1137/0216030.

[38] R. Tamassia and I. G. Tollis. Planar grid embeddings in linear time. *IEEE Transactions on Circuits and Systems*, 36(9):1230–1234, 1989. doi:10.1109/31.34669.

[39] L. Vismara, G. Di Battista, A. Garg, G. Liotta, R. Tamassia, and F. Vargiu. Experimental studies on graph drawing algorithms. *Software: Practice and Experience*, 30(11):1235–1284, 2000. doi:10.1002/1097-024X(200009)30:11$<$1235::AID-SPE339$>$3.0.CO$;$2-B.