A Walk on the Wild Side: A Shape-First Methodology for Orthogonal Drawings

Giordano Andreola
Roma Tre University, Italy

Susanna Caroppo
Roma Tre University, Italy

Giuseppe Di Battista
Roma Tre University, Italy

Fabrizio Grosso
CeDiPa - University of Perugia, Italy

Maurizio Patrignani
Roma Tre University, Italy

Allegra Strippoli
Roma Tre University, Italy

Abstract

Several algorithms for the construction of orthogonal drawings of graphs, including those based on the Topology-Shape-Metrics (TSM) paradigm, tend to prioritize the minimization of crossings. This emphasis has two notable side effects: some edges are drawn with unnecessarily long sequences of segments and bends, and the overall drawing area may become excessively large. As a result, the produced drawings often lack geometric uniformity. Moreover, orthogonal crossings are known to have a limited impact on readability, suggesting that crossing minimization may not always be the optimal goal. In this paper, we introduce a methodology that "subverts" the traditional TSM pipeline by focusing on minimizing bends. Given a graph G, we ideally seek to construct a rectilinear drawing of G, that is, an orthogonal drawing with no bends. When not possible, we incrementally subdivide the edges of G by introducing dummy vertices that will (possibly) correspond to bends in the final drawing. This process continues until a rectilinear drawing of a subdivision of the graph is found, after which the final coordinates are computed. We tackle the (NP-complete) rectilinear drawability problem by encoding it as a SAT formula and solving it with state-of-the-art SAT solvers. If the SAT formula is unsatisfiable, we use the solver's proof to determine which edge to subdivide. Our implementation, DOMUS, which is fairly simple, is evaluated through extensive experiments on small- to medium-sized graphs. The results show that it consistently outperforms OGDF 's TSM-based approach across most standard graph drawing metrics.

2012 ACM Subject Classification Theory of computation → Computational geometry; Theory of computation → Design and analysis of algorithms; Mathematics of computing → Graph algorithms

Keywords and phrases Non-planar Orthogonal Drawings, SAT Solver, Experimental Comparison

Digital Object Identifier 10.4230/LIPIcs.GD.2025.35

Related Version *Full Version*: `https://arxiv.org/abs/2508.19416` [1]

Funding This research was supported, in part, by MUR of Italy (PRIN Project no. 2022ME9Z78 – NextGRAAL and PRIN Project no. 2022TS4Y3N – EXPAND). The fourth author was supported by Ce.Di.Pa. - PNC Programma unitario di interventi per le aree del terremoto del 2009-2016 - Linea di intervento 1 sub-misura B4 - "Centri di ricerca per l'innovazione" CUP J37G22000140001.

1 Introduction

Orthogonal drawings, where edges are represented as chains of horizontal and vertical segments, are a foundational topic in Graph Drawing, valued both for their practical applications and for the rich body of research they have inspired over the past four decades.

The most well-known and widely studied methodology for constructing orthogonal drawings is the *Topology-Shape-Metrics* (*TSM*) approach (see, e.g., [19, 42, 43]). TSM generates an orthogonal drawing in three steps. First, a planar embedding (*topology*) of the graph is computed, and *dummy vertices* are possibly introduced to represent edge crossings. Second, a *shape* – minimizing the number of *bends* (points where two edge segments meet at a right angle) – is computed, see [40]. Third, an orthogonal drawing (*metrics*) consistent with the computed shape is constructed. Although originally designed for graphs with maximum degree 4, TSM has been extended to general graphs by either representing vertices as boxes sized proportionally to their degree (see, e.g., [4, 5]), or by allowing partial edge overlaps, which requires adapting the flow-based optimization accordingly (see, e.g., [22, 24]).

While TSM is arguably the most well-known method, alternative approaches have been proposed for orthogonal drawings (see, e.g., [10, 11, 33, 34, 39]). Nonetheless, the comprehensive and still influential quantitative aggregate [3] evaluation in [20] shows that the TSM approach outperforms all the above cited alternative methods across nearly all standard graph drawing metrics. Note that other algorithms (see e.g., [28]) have been proposed and assessed not through quantitative measures but via user studies. Also, the algorithm in [26] neglects the experiments in [20] in favor of a comparison with [28].

Furthermore, the success of TSM is underscored by its implementation in widely used open-source libraries such as OGDF [13] and GDToolkit [18], as well as its adoption in the Graph Drawing engine of yWorks [22, 24, 29, 36], a leading company in graph visualization.

However, two key observations can be made. (1) Existing algorithms, including those based on TSM, prioritize minimizing crossings. This often leads to unnecessarily long edge paths with many bends and results in drawings with large area and poor geometric uniformity. (2) Several studies (e.g., [27]) have shown that orthogonal crossings, where edges meet at right angles, have minimal impact on readability. This suggests that aggressively minimizing crossings in orthogonal drawings may not always be the optimal design goal.

In this paper, we start from the two aforementioned observations and introduce a methodology, called *Shape-Metrics* (*SM*), that "subverts" the traditional TSM pipeline. SM focuses first on bends. Namely, given a graph G, we ideally seek to construct a *rectilinear drawing* of G, that is, a drawing in which all edges are represented by straight horizontal or vertical segments with no bends. When such a drawing is not feasible, we incrementally subdivide the edges of G by introducing dummy vertices. Each of these vertices will (possibly) correspond to a bend in the final drawing. We continue this process until we obtain a subdivision of G that admits a rectilinear drawing. Once such a subdivision is found, the final metric phase computes the actual coordinates of the drawing.

SM is based on the concept of *shaped* graph, which we define as an assignment of a label from the set $\mathcal{L} = \{L, R, D, U\}$ to each of its edges, indicating the direction – Left, Right, Down, or Up – of that edge in a drawing. Given a shaped graph, it can be decided in polynomial time [30] if it admits a rectilinear drawing with that shape. On the other hand, checking if a graph admits a rectilinear drawable shape is NP-complete [21]. We exploit a new simple characterization of rectilinear drawable shaped graphs. Informally, a shaped cycle is said to be *complete* if its edges have all the four labels; we show that a shaped graph is rectilinear drawable if and only if all its cycles are complete.

We tackle the NP-hardness barrier by formulating the rectilinear drawability testing problem as a Boolean satisfiability (SAT) problem and leveraging the power of modern SAT solvers. SAT solvers have already proven to be effective in combinatorics and geometry (see, e.g., [12, 17, 31, 37, 38]). However, as far as we know, SAT solvers have rarely been applied to Graph Drawing. The related works we are aware of are [9], which employs a *hybrid* ILP/SAT approach to compute *st*-orientations and visibility representations, [8] where a SAT solver is used to compute book embeddings, and [14, 15, 16] where a SAT solver is used to test upward planarity.

There are differences and analogies between SM and TSM. TSM leverages the fact that checking if a graph is planar can be done efficiently. On our side we have that, unfortunately, checking if a graph is rectilinear drawable is NP-complete [21]. On the other hand both methodologies use specific properties of shapes: TSM's cornerstone is the balance of turns in shapes of faces of planar graphs [44], while SM's cornerstone is the completeness of cycles. Also, both methodologies augment the graph to be drawn with dummy vertices, TSM to represent crossings, SM to represent bends.

We implemented SM in a tool called DOMUS (*Drawing Orthogonal Metrics Using the Shape*) and experimented its effectiveness against the TSM implementation available in OGDF [13] measuring widely adopted Graph Drawing metrics [3, 20]. Although other comparable tools exist (e.g., GDToolkit [18]), we selected OGDF as our benchmark due to its open-source nature, widespread adoption within the Graph Drawing community, and the fact that it has benefited from over a decade of active development and optimization. In view of the results in [20] it was pointless to compare DOMUS with the algorithms in [10, 11, 33, 34, 39]. Also, the algorithms in [26, 28] adopt a definition of orthogonal drawing which is different from the widely adopted one given in this paper. E.g., even if a vertex has degree less or equal than 4 more than one of its incident edges can exit from the same direction. Hence, comparing DOMUS with them would be unfair.

We did two sets of experiments, which we refer to as "in-vitro" and "in-the-wild". The in-vitro experiments aim to thoroughly evaluate the performance and characteristics of SM, including the use of the SAT solver, on a dataset of random graphs with predefined sizes and densities. These graphs have a maximum degree of 4, which is the baseline for orthogonal drawings, and contain $20-60$ vertices, a range considered important for node-link representations in graph visualization (see, e.g., [25]). The in-the-wild experiments use the Rome Graphs dataset [20], which is one of the most popular collections of graphs [3], comprised of $11,534$ graphs, having $10-100$ vertices and $9-158$ edges, with maximum degree 13. The experiments show that for most of the typical graph drawing metrics [3], DOMUS outperforms the TSM algorithm implemented in OGDF. Moreover, SM is simple to implement: it reduces shape constraints to a SAT formula, invokes an off-the-shelf SAT solver, and computes the final drawing metrics using a slight variation of standard TSM compaction techniques. Figure 1 shows two drawings of the same graph, one produced with OGDF and the other with DOMUS.

The paper is organized as follows. Preliminaries are in Section 2. Section 3 illustrates the characterization of rectilinear drawable shaped graphs which is the milestone of SM. Section 4 describes SM, clarifying how its steps are activated and implemented and the role played by the SAT-solver. Section 5 and Section 6 show the in-vitro and the in-the-wild experiments, respectively. Conclusions and open problems are proposed in Section 7. Because of space limitations some proofs are only sketched. A full version of such proofs and other extra material is in [1].

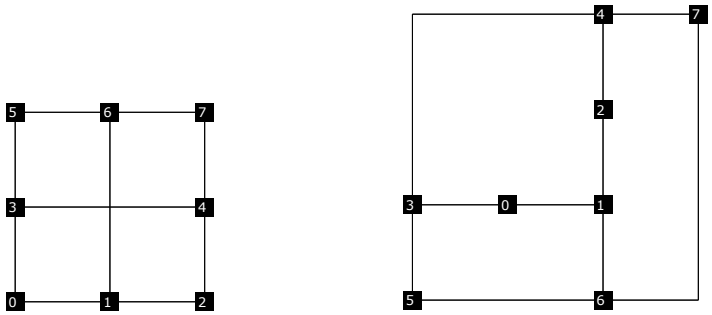


Figure 1 The graph in Fig. 2.4 of [19], represented: **(a)** with DOMUS and **(b)** with OGDF.

2 Preliminaries

For basic graph drawing terminology and definitions refer, e.g., to [19, 32].

Let G be a graph. with vertex set $V(G)$ and edge set $E(G)$. For technical reasons, we assign an arbitrary (e.g., random) orientation to each edge in $E(G)$; thus, if an edge is denoted as (u, v), it is considered as directed from u to v. However, when referring to paths, cycles, vertex degrees, or connectivity in G, we disregard these orientations. A path (resp., cycle) of G is *simple* if each of its vertices occurs only once.

A *cut-vertex* is a vertex whose removal disconnects G. Graph G is *biconnected* if it has no cut-vertex. A *biconnected component* of G is a maximal (in terms of vertices and edges) biconnected subgraph of G. A biconnected component is *trivial* if it consists of a single edge and *non-trivial* otherwise. Unless otherwise specified we assume that graphs are connected.

In a *rectilinear drawing* of G: **(i)** Each vertex $v \in V(G)$ is represented by a point with integer coordinates, denoted by $x(v)$ and $y(v)$. **(ii)** No two vertices are represented by the same point. **(iii)** Each edge is represented by either a horizontal or a vertical line segment connecting the points that represent its end-vertices. **(iv)** The interior of a segment representing an edge does not intersect any point representing a vertex. A graph is said to be *rectilinear drawable* if it admits a rectilinear drawing. Note that such a drawing requires each vertex to have degree at most 4. Hence, unless otherwise specified, we will consider graphs with maximum vertex degree 4.

We use the label set $\mathcal{L} = \{L, R, D, U\}$ to denote directions, where L, R, D, and U stand for left, right, down, and up, respectively. Also, we say that L and R are *opposite* labels, and likewise, D and U are *opposite* labels. For any label $A \in \mathcal{L}$, we denote its opposite by $\overline{A}$.

A *shape* of a graph G is a function λ that assigns to each edge $(u, v) \in E(G)$ a label from the set $\mathcal{L}$. Consider an edge (u, v). Labeling (u, v) with, say, R indicates that in the shape of G, the edge is directed to the right when traversed from u to v. Clearly, if the edge is traversed in the opposite direction (from v to u) the direction should be left. Hence, with a little abuse of notation, if $\lambda(u, v) = R$ (resp. $\lambda(u, v) = L$) we say that $\lambda(v, u) = L$ (resp. $\lambda(v, u) = R$). The other labels are treated analogously. Additionally, for each pair of edges that share a vertex v it is mandatory that $\lambda(v, u) \neq \lambda(v, w)$. This constraint ensures that two edges sharing the same vertex "cannot point in the same direction", preventing the two edges from "overlapping". We define a *shaped graph* to be a graph with a shape.

A shape can be assigned to a graph by starting from one of its rectilinear drawings, if such a drawing exists. Namely, given a rectilinear drawing Γ of a graph G we can label the edges $E(G)$ as follows. For each $(u, v) \in E(G)$: **(i)** If $x(u) < x(v)$ we set $\lambda(u, v) = R$, **(ii)** If

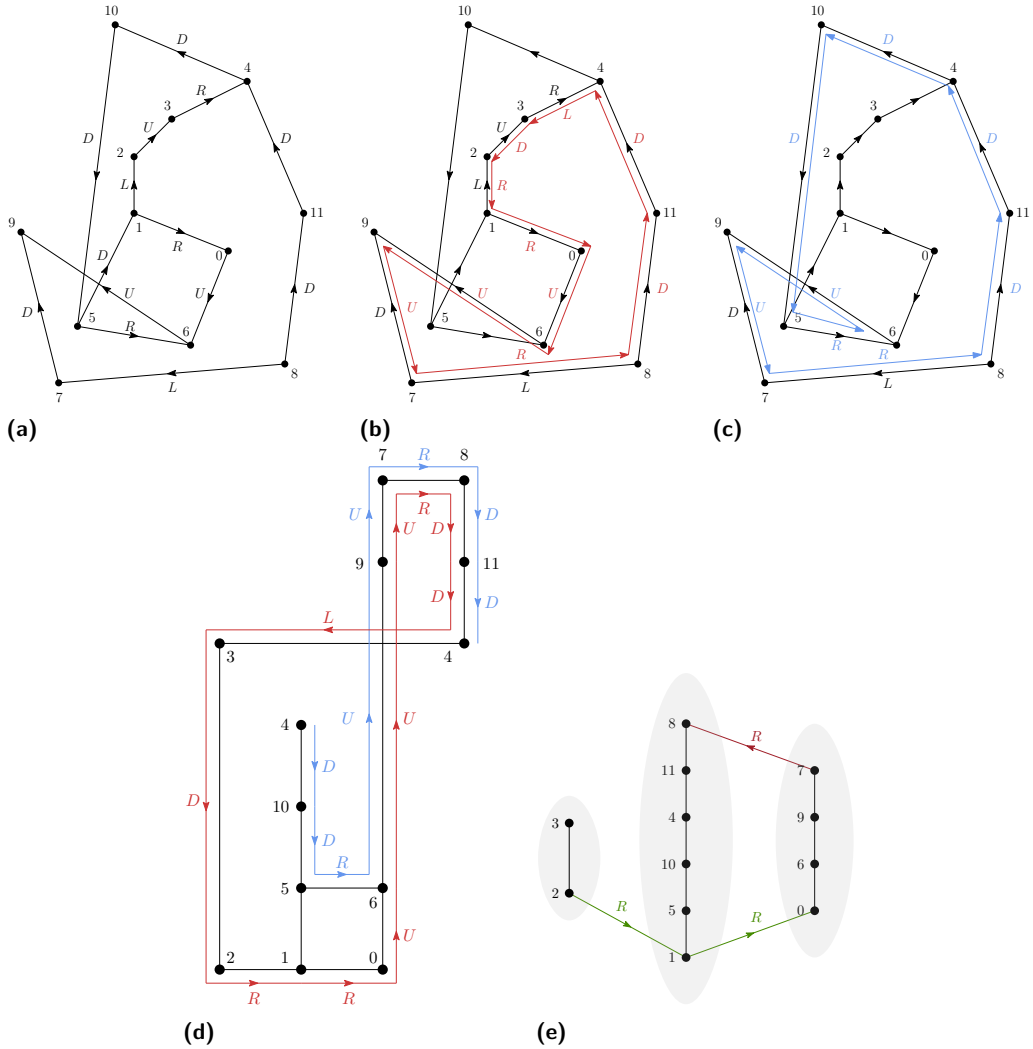


Figure 2 **(a)** A shaped graph G. **(b)** A complete cycle of G. **(c)** Another cycle of G. **(d)** A failed attempt to draw G according to its shape. **(e)** Graph G_x for the shaped graph of Figure 2a.

$x(u) > x(v)$ we set $\lambda(u, v) = L$, **(iii)** If $y(u) < y(v)$ we set $\lambda(u, v) = U$, **(iv)** If $y(u) > y(v)$ we set $\lambda(u, v) = D$. We say that this is the *shape* of Γ. It is easy to check that this labeling is consistent with the definition of shape, and hence it is also a valid shape for G.

A shaped graph is *rectilinear drawable* if it admits a rectilinear drawing having exactly its shape. The shaped graph in Figure 2a is not rectilinear drawable. Figure 2d shows a failed attempt to construct a rectilinear drawing of such a shaped graph.

Given a graph G a *subdivision* of G is a graph G' obtained by replacing certain edges of $E(G)$ with internally vertex-disjoint simple paths. The new vertices in $V(G') \setminus V(G)$ introduced along these paths are called *dummy vertices*. An *orthogonal drawing* Γ of G is a rectilinear drawing of a subdivision of G. Let v be a dummy vertex, if the two segments incident to v in Γ are one horizontal and one vertical, then v is called a *bend*.

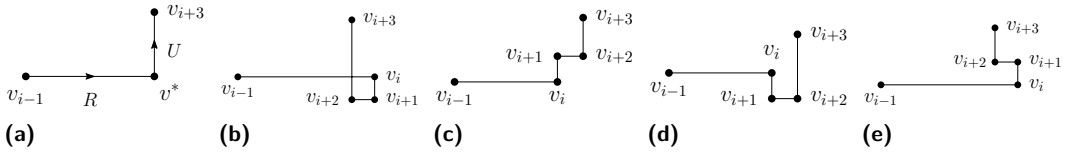


Figure 3 Illustration for the proof of Theorem 1. **(a)** The partial rectilinear drawing of c' according to the new sequence. The result of the extension of the drawing of c' to c by re-inserting labels **(b)** D and L; **(c)** U and R; **(d)** D and R; and **(e)** U and L.

3 Rectilinear Drawable Shapes and the Shape of Cycles

The methodology we present in Section 4 is based on the results presented in this section.

Consider a shaped simple cycle $c = v_0, v_1, \ldots, v_{p-1}$, with $p \geq 4$, of G. We say that c is *complete* if it contains four pairs of vertices v_i, v_{i+1}, v_j, v_{j+1}, v_h, v_{h+1}, and v_k, v_{k+1} such that: **(i)** $\lambda(v_i, v_{i+1}) = L$, **(ii)** $\lambda(v_j, v_{j+1}) = R$, **(iii)** $\lambda(v_h, v_{h+1}) = D$, **(iv)** $\lambda(v_k, v_{k+1}) = U$, where the vertices indexes of c are intended mod p. An example of a complete cycle is shown in Figure 2b, while Figure 2c illustrates a cycle that is not complete.

▶ **Theorem 1.** *A shaped simple cycle $c = v_0, v_1, \ldots, v_{p-1}$ with $p \geq 4$ is rectilinear drawable if and only if it is complete.*

Sketch of Proof. If c admits a rectilinear drawing, then intersecting it with horizontal and vertical lines (avoiding vertices) reveals edges labeled U, D, L, and R. Hence, c is complete. Conversely, suppose c is complete. We prove that it admits a rectilinear drawing by induction on the number of vertices p. For $p = 4$, the completeness implies the presence of all directions in $\{L, R, U, D\}$. The only valid labelings without consecutive opposites are R–U–L–D and R–D–L–U, which form rectangles. Now, assume that any complete cycle with $p - 2$ vertices ($p \geq 6$) is drawable. Let c be a complete cycle with $p \geq 6$ vertices. Without loss of generality, assume no two consecutive labels are equal. Then, the label sequence $\Lambda = s_0, \ldots, s_{p-1}$ alternates between $\{L, R\}$ and $\{U, D\}$, contains all four labels, and avoids consecutive equal or opposite pairs. Then, there exists a removable pair s_i, s_{i+1} such that the resulting cycle c' remains complete. By induction, c' is drawable. Denote (see Figure 3) by (v_{i-1}, v^*) (resp., by (v^*, v_{i+3})) the edge of c' corresponding to s_{i-1} (resp., s_{i+2}), where v^* represents the vertices v_i, v_{i+1} and v_{i+2} that have been merged together from the removal of the two edges. Reinserting s_i, s_{i+1} as a right-angle bend (e.g., $\langle U, R \rangle$, $\langle D, L \rangle$) and applying simple geometric adjustments yields a drawing of c. Thus, c is rectilinear drawable. ◀

To test if a shaped graph is rectilinear drawable we can use the following theorem, which has been proved in [30].

▶ **Theorem 2** ([30, Theorem 3]). *Given a shaped graph G, there exists an $O(|V(G)| \cdot |E(G)|)$ algorithm to test if it is rectilinear drawable. In the positive case a rectilinear drawing can be found in $O(|V(G)| \cdot |E(G)|)$ time.*

The proof in [30] is based on constructing, starting from the given shaped graph, two auxiliary directed graphs which we call G_x and G_y. For the sake of readability, we call the vertices and the edges of such graphs *nodes* and *arcs*, respectively.

We say that two vertices v_0 and v_{k-1} of G are *x-aligned* if a path $v_0, \ldots, v_{k-1}$ exists in G such that $\lambda(v_i, v_{i+1}) = D$ $(i = 0, \ldots, k-2)$. Also, we say that a vertex of G is *x-aligned* with itself. Graph G_x is defined as follows. A node of G_x is a maximal set of x-aligned vertices of G. Given two nodes μ and ν of G_x, by the definition of shape, they are disjoint sets. There

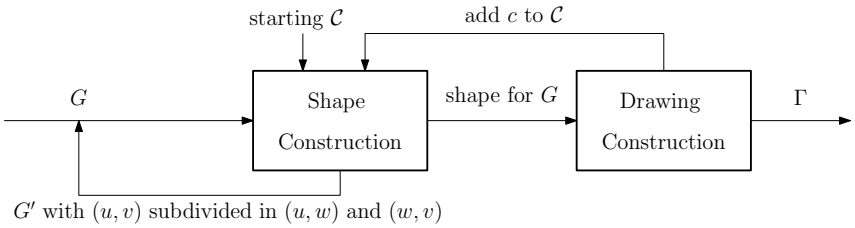


Figure 4 SM: a methodology for constructing orthogonal drawings of graphs.

is an arc from node μ to node ν if there are two vertices u and v of G such that $u \in \mu$, $v \in \nu$, and either $(u, v) \in E$, and $\lambda(u, v) = R$ or $(v, u) \in E$, and $\lambda(v, u) = L$. Figure 2e shows graph G_x for the shaped graph in Figure 2a. Graph G_y is defined analogously.

In [30] it is proved that if there is an arc in G_x (resp., G_y) from μ to ν, then in any rectilinear drawing of G the vertices in μ should be placed to the left of (resp., above) the vertices in ν. Also all the vertices in the same node should be vertically (resp., horizontally) aligned. This implies that a necessary condition for the shape to be rectilinear drawable is that both G_x and G_y are acyclic. Observe that graph G_x of Figure 2e is cyclic and the shaped graph in Figure 2a is not rectilinear drawable. The authors show that this condition is also sufficient by constructing a drawing. Namely, they assign the x-coordinates to the vertices of $V(G)$ as follows: (1) the same x coordinate is assigned to all the vertices corresponding to the same node of G_x, (2) different coordinates are assigned to vertices corresponding to different nodes of G_x, and (3) x-coordinates increase according to the orientations of the nodes of G_x. The y-coordinates are assigned analogously. Since in our case the vertices of $V(G)$ have degree at most 4, it easy to see that the construction of G_x and G_y, the acyclicity test, and, in case, the construction of the drawing can be accomplished in $O(|V(G)|)$ time.

We are now able to prove the following characterization.

▶ **Theorem 3.** *A shaped graph G is rectilinear drawable if and only if all its simple cycles are complete. In case G is not rectilinear drawable, a simple cycle of G which is not complete can be found in $O(|V(G)|)$ time.*

Sketch of Proof. The necessity follows from Theorem 1. To prove the sufficiency consider a shaped graph G and its auxiliary directed graphs G_x and G_y. From the proof of Theorem 2 we have that the presence of a cycle in G_x or G_y implies that G is not rectilinear drawable using the prescribed shape. We prove that a cycle in G_x or in G_y implies a non-complete simple cycle in G. Let $C_x = \mu_0, \ldots, \mu_{p-1}$ be a cycle in G_x, possibly p=1. See Figure 2e, where $p = 2$. Consider the p pairs u_i, v_{i+1} of vertices of G, such that $u_i \in \mu_i$ and $v_{i+1} \in \mu_{i+1}$, $(u_i, v_{i+1}) \in E(G)$, and $\lambda(u_i, v_{i+1}) = R$, for $i \in \{0, \ldots, p-1\}$ (where indices are taken mod p). By the construction of G_x, there exists a path Π_i in G from v_i to u_i that contains edges all labeled D or all labeled U. The concatenation of the paths and edges $\Pi_0, (u_0, v_1), \Pi_1, \ldots, (u_{p-2}, v_{p-1}), \Pi_{p-1}, (u_{p-1}, v_0)$ forms a simple cycle c in G that is not complete. The cycle of Figure 2e contains vertices 8, 11, 4, 10, 5, 1, 0, 6, 9, and 7. The construction of G_x and of G_y, the acyclicity test, the search of a cycle C_x, and the computation of c can all be done in $O(|V(G)|)$ time. ◀

4 A Methodology for Constructing Orthogonal Drawings of Graphs

This section presents SM. Given an n-vertex graph G as input, it produces an orthogonal drawing Γ of G as output. SM consists of two steps, called *Shape Construction* and *Drawing Construction*, see Figure 4. Each of these steps can be repeated multiple times.

4.1 Shape Construction

We know from Theorem 3 that G admits a rectilinear drawable shape if and only if it admits a shape in which all simple cycles are complete. Consequently, the Shape Construction step may attempt to compute a shape for G that enforces completeness for all simple cycles. However, since the number of simple cycles in G can be exponential in the size of G, checking all of them is computationally infeasible. To address this, the first phase of Shape Construction, called *Cycle Selection*, selects a subset $\mathcal{C}$ of simple cycles in G so that the Shape Construction searches for a shape that ensures completeness only for the cycles in $\mathcal{C}$, with the hope that this will be enough to enforce completeness for all cycles.

If no such shape exists, this implies that a rectilinear drawable shape for G does not exist–since adding more constraints cannot make a previously unsolvable problem solvable. In this case, the Shape Construction step attempts to identify an edge (u, v) that is "responsible" for the infeasibility, and splits it into two edges, (u, w) and (w, v), by introducing a dummy vertex w. It then restarts the process on the modified graph G', which is a subdivision of G. The vertex w will possibly be drawn as a bend in the final orthogonal drawing. Additionally, splitting the edge (u, v) indirectly alters the cycle set $\mathcal{C}$, producing a new set $\mathcal{C}'$, as the new vertex w is now included in all cycles of $\mathcal{C}$ that originally contained (u, v).

If such a shape exists, this does not necessarily imply that it is rectilinear drawable. Thus, the shape is passed as input to the Drawing Construction step for further verifications.

4.2 Drawing Construction

The Drawing Construction step relies on two theorems. Theorem 2, states that a shape can be efficiently tested for rectilinear drawability, and if the test is positive, a rectilinear drawing can be constructed efficiently. Theorem 3, states that if a shape is not rectilinear drawable a cycle c that is not complete can be efficiently identified. Hence, the Drawing Construction step either produces Γ directly or suggests to the Shape Construction step to add c to the cycle set $\mathcal{C}$, selectively enlarging the set of cycles considered for completeness.

Observe that, in general, due to the possible introduction of dummy vertices during the Shape Construction step, the resulting rectilinear drawing Γ is the rectilinear drawing of a subdivision of G. Hence, Γ is an orthogonal drawing of G.

4.3 Implementing the Shape and the Drawing Construction

Implementing the Shape Construction step essentially involves selecting an initial cycle set $\mathcal{C}$ and checking whether a shape of G exists that enforces completeness for all cycles in $\mathcal{C}$.

The selection of $\mathcal{C}$ can be done in various ways. A natural and intuitive strategy is to include cycles that cover all edges belonging to non-trivial biconnected components of G. Further details are given in Section 5.

The checking for the existence of a shape is the most important part of the Shape Construction. We do this by converting completeness constraints into a CNF Boolean formula, $\mathcal{F}_{G,\mathcal{C}}$ and then by using a SAT solver to check satisfiability. This method benefits from (1) the efficiency of modern SAT solvers and (2) the solver's ability to provide a proof if unsatisfiable. From this proof, we identify an edge label variable causing unsatisfiability, indicating the edge (u, v) is over constrained, so we split it by adding a dummy vertex.

Formula $\mathcal{F}_{G,\mathcal{C}}$ has four boolean variables for each edge (v, w) of $E(G)$. Such variables state if $\lambda(v, w)$ is equal to L, R, D, or U. It also has three types of clauses to state that: (1) each edge is assigned to exactly one label, (2) no vertex has two adjacent edges with the same direction, and (3) each simple cycle in $\mathcal{C}$ is complete.

The full formal description of $\mathcal{F}_{G,\mathcal{C}}$ is as follows. Formula $\mathcal{F}_{G,\mathcal{C}}$ has four boolean variables for each edge (v, w) of $E(G)$. We call them $\ell_{v,w}$, $r_{v,w}$, $d_{v,w}$, and $u_{v,w}$. We have that:

1. $\ell_{v,w} = true$ iff $\lambda(v, w) = L$,
2. $r_{v,w} = true$ iff $\lambda(v, w) = R$,
3. $d_{v,w} = true$ iff $\lambda(v, w) = D$, and
4. $u_{v,w} = true$ iff $\lambda(v, w) = U$

To simplify the description, we also introduce the variable $\ell_{w,v}$ (resp. $r_{w,v}$), which is equal to $r_{v,w}$ (resp. $\ell_{v,w}$). Analogously, we introduce the variable $d_{w,v}$ (resp. $u_{w,v}$), which is equal to $u_{v,w}$ (resp. $d_{v,w}$).

Formula $\mathcal{F}_{G,\mathcal{C}}$ has three sets of clauses.

The first set of clauses ensures that each edge (v, w) is assigned exactly one label. Namely, for each edge (v, w) we have the following clauses: **(i)** $(\ell_{v,w} \vee r_{v,w} \vee d_{v,w} \vee u_{v,w})$, **(ii)** $(\neg\ell_{v,w} \vee \neg r_{v,w})$, **(iii)** $(\neg d_{v,w} \vee \neg\ell_{v,w})$, **(iv)** $(\neg d_{v,w} \vee \neg r_{v,w})$, **(v)** $(\neg u_{v,w} \vee \neg\ell_{v,w})$, **(vi)** $(\neg u_{v,w} \vee \neg r_{v,w})$, **(vii)** $(\neg u_{v,w} \vee \neg d_{v,w})$. Clause **(i)** is true if at least one label is assigned to (v, w), clauses **(ii)**–**(vii)** are true if at most one label is assigned to (v, w).

The second set of clauses guarantees that for each vertex v with at least two neighbors, there are no two of such neighbors w' and w'' of v such that $\lambda(v, w') = \lambda(v, w'')$. Let $u_0, \ldots, u_k$ be the neighbors of v, with $k = 1, 2, 3$ according to the degree of v minus one. We distinguish two cases. If v has degree 4, we have 4 clauses, $(\ell_{v,u_0} \vee \ell_{v,u_1} \vee \ell_{v,u_2} \vee \ell_{v,u_3})$, $(r_{v,u_0} \vee r_{v,u_1} \vee r_{v,u_2} \vee r_{v,u_3})$, $(d_{v,u_0} \vee d_{v,u_1} \vee d_{v,u_2} \vee d_{v,u_3})$, and $(u_{v,u_0} \vee u_{v,u_1} \vee u_{v,u_2} \vee u_{v,u_3})$, that are satisfied if for each label $l \in \mathcal{L}$ there exists an index i $(i = 0, \ldots, 3)$ such that $\lambda(v, u_i) = l$. Otherwise (v has degree 2 or 3), for each pair of neighbors u_i and u_j of v, we have $(\neg\ell_{v,u_i} \vee \neg\ell_{v,u_j})$, $(\neg r_{v,u_i} \vee \neg r_{v,u_j})$, $(\neg d_{v,u_i} \vee \neg d_{v,u_j})$, and $(\neg u_{v,u_i} \vee \neg u_{v,u_j})$.

The third set of clauses guarantees that all simple cycles in $\mathcal{C}$ are complete. For each simple cycle $v_0, v_1, \ldots, v_{n-1}$ of $\mathcal{C}$ we impose that it contains all labels in $\mathcal{L}$ while traversing it. This is done with four clauses as follows: $(\ell_{v_0,v_1} \vee \ell_{v_1,v_2} \vee \cdots \vee \ell_{v_{n-2},v_{n-1}} \vee \ell_{v_{n-1},v_0})$, $(r_{v_0,v_1} \vee r_{v_1,v_2} \vee \cdots \vee r_{v_{n-2},v_{n-1}} \vee r_{v_{n-1},v_0})$, $(d_{v_0,v_1} \vee d_{v_1,v_2} \vee \cdots \vee d_{v_{n-2},v_{n-1}} \vee d_{v_{n-1},v_0})$, and $(u_{v_0,v_1} \vee u_{v_1,v_2} \vee \cdots \vee u_{v_{n-2},v_{n-1}} \vee u_{v_{n-1},v_0})$

It is easy to see that $\mathcal{F}_{G,\mathcal{C}}$ is satisfiable if and only if G has a shape such that all the cycles in $\mathcal{C}$ are complete.

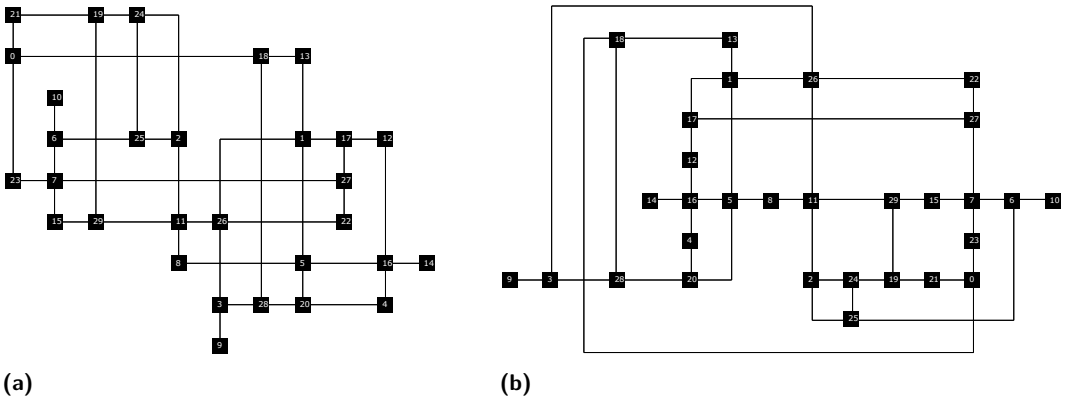
Implementing the Drawing Construction step primarily involves building the two auxiliary graphs G_x and G_y and checking each for acyclicity. If either graph contains a cycle, it can be used to identify a corresponding cycle c in G that is not complete.

Finally, we observe that the process described in the methodology is guaranteed to terminate, since an orthogonal drawing of a graph always exists (see, e.g, [10, 11]).

5 An Experimental Evaluation on Maximum Degree 4 Graphs

We implemented SM in our tool DOMUS and evaluate its effectiveness through comprehensive "in vitro" experiments conducted using DOMUS. Specifically, we describe: **(i)** the implementation choices in DOMUS that define some aspects of SM, **(ii)** the benchmark adversary for comparison, **(iii)** the graph dataset and its key characteristics, **(iv)** the computational platform used, **(v)** the evaluation metrics, **(vi)** the experimental results for each metric, and **(vii)** internal performance indicators, such as SAT solver invocations per graph.

Two drawings of a graph in the dataset are in Figure 5. **(i)** In DOMUS, we made specific implementation choices regarding the SAT solver, the initial cycle set $\mathcal{C}$, coordinate computation, and randomization. For the SAT solver, we chose Glucose [2] because it is open source and it is well known for its efficiency. Importantly, Glucose provides proofs of



(a) **(b)**

Figure 5 A graph in the dataset: **(a)** drawn by DOMUS and **(b)** drawn by OGDF.

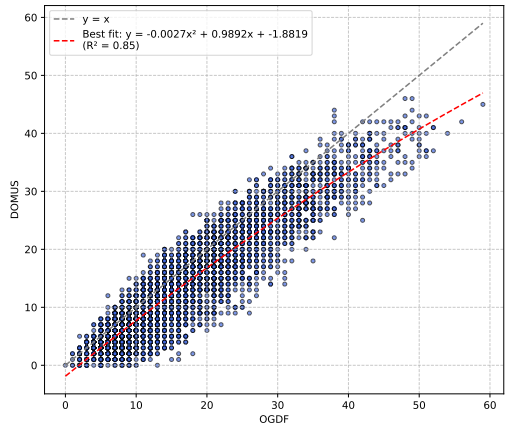
unsatisfiability, which we use to identify edges to split, thereby increasing flexibility in the drawing. To initialize the cycle set $\mathcal{C}$, DOMUS computes a *cycle basis* covering all edges in the non-trivial biconnected components of the input graph G. This is done via a BFS from an arbitrary root to construct a tree T; for each non-tree edge (u, v), we add the cycle formed by (u, v) and the unique paths from u and v to their lowest common ancestor in T. This results in an initial set of size $|\mathcal{C}| = |E(G)| - |V(G)| + 1$. For coordinate assignment, after computing G_x and G_y, DOMUS uses a compaction algorithm similar to that in OGDF. For details on compaction algorithms see, e.g., [6, 7, 29, 35]. Finally, Glucose can make random choices. In order to enforce replicability of the experiments we forced it to work deterministically.

(ii) For the motivations discussed in Section 1, we compared the performance of DOMUS against the TSM implementation available in OGDF [13]. Another option was to use as a benchmark one of the tools (see, e.g. [28]) that have been evaluated via user studies. However, this comparison would be useless, since most of this tools adopt a definition of orthogonal drawing which is different from the widely adopted one given in Section 2. E.g., even if a vertex has degree less or equal than 4 more than one of its incident edges can exit from the same direction. Also, vertices and bends neither have integer coordinates nor it is easy to "snap" them to a grid (see [1] for a comparison with these models).

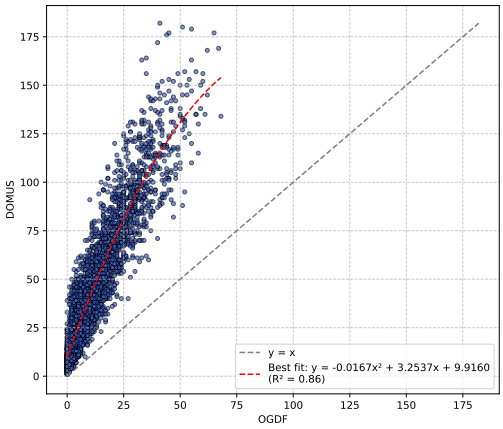
(iii) We generated 4,100 connected graphs uniformly at random, each with maximum vertex degree 4. Their sizes range from 20 to 60 vertices, and their densities – defined as the ratio between the number of edges to the number of vertices – span 1.25 to 1.75 in steps of 0.005. For degree 4 graphs, the theoretical maximum density is 2. Namely, for each $n = 20, \ldots, 60$ and $i = 1, 2, \ldots, 100$ we generated a graph $G_{n,i}$ with density $d_i = 1.25 + i \cdot (1.75 - 1.25)/100$. Hence, for each value of n the graphs have roughly 100 possible densities ranging from 1.25 to 1.75, in increments of 0.005. In terms of number of edges we have $|E(G_{n,i})| = \lfloor n \cdot d_i \rfloor$. For instance, the graph $G_{20,50}$ has 30 edges, while $G_{60,100}$ has 105 edges. For each n and d_i we initialized a graph with n vertices and no edges. Then, we repeatedly randomized two vertices u and v of the graph and added an edge between them, until d_i was reached. If $u = v$ or if (u, v) was already in the graph, or if either u or v was already degree 4, we skipped the pair. Finally, the instance was rejected if non-connected.

Conducting experiments on a set of randomly generated graphs was a necessary choice, as, to the best of our knowledge, no publicly available collection of maximum-degree-4 graphs is currently recognized as a standard benchmark for graph drawing experiments.

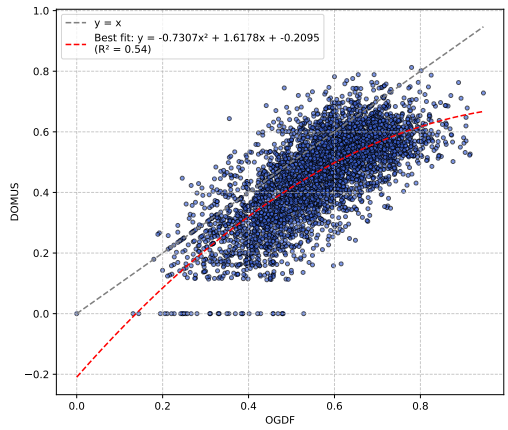
(iv) All the experiments were performed running OGDF and DOMUS on a personal computer with a 3.15 GHz Intel processor (comparable with an M1 Apple processor).



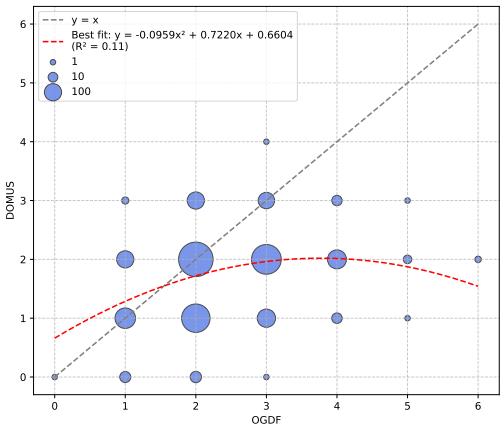
(a) Bends.



(b) Crossings.



(c) Bends Deviation.

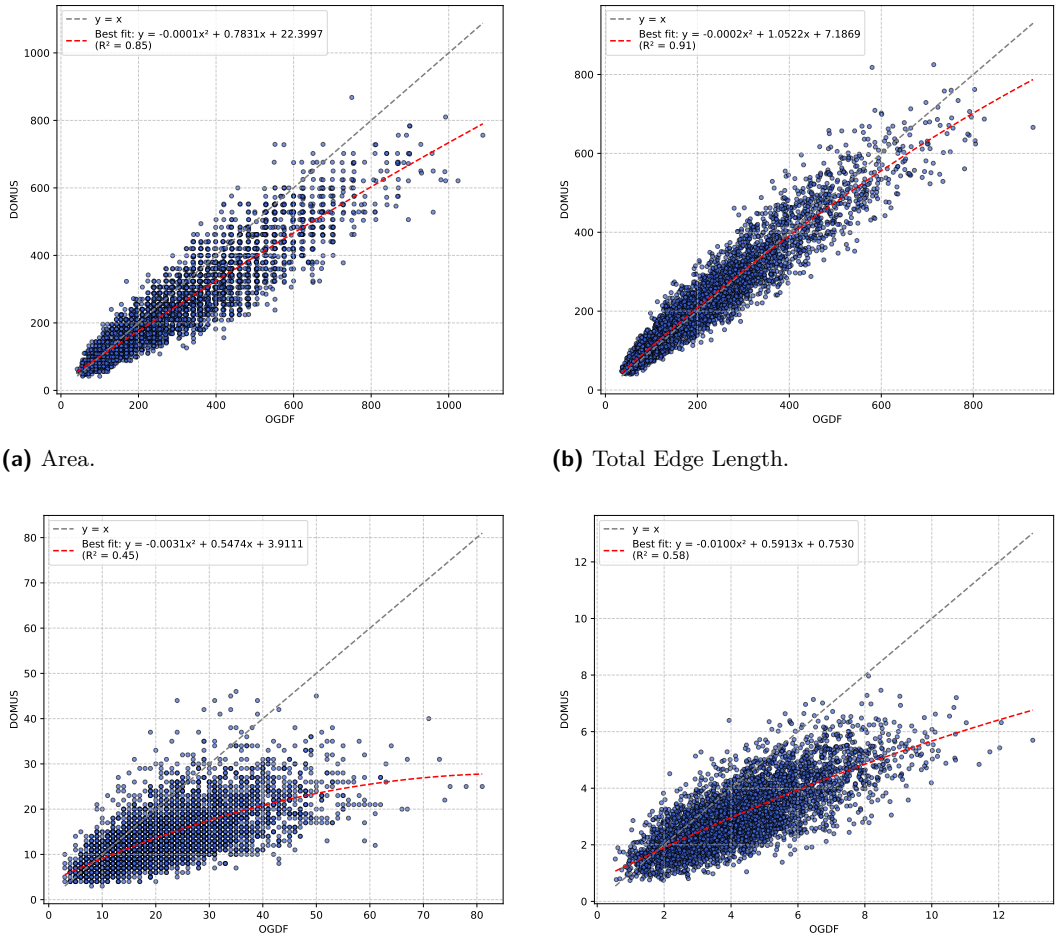


(d) Max Bends.

Figure 6 Effectiveness of OGDF and DOMUS "in vitro": Bends and Crossings.

(v) Following the framework proposed in [20], we compared the drawings produced by DOMUS and those generated by OGDF using the following metrics: the total number of **Bends** in the drawing; the total number of edge **Crossings**; the standard deviation of the number of bends per edge (**Bends Deviation**); the maximum number of bends on any single edge (**Max Bends**); the **Area** occupied by the drawing; the sum of the length of all the edges (**Total Edge Length**); the length of the longest edge (**Max Edge Length**); the standard deviation of edge lengths (**Edge Length Deviation**) and the total computation **Time** to build the drawing, measured in seconds. Using the terminology in [3], we adopt most of the metrics suggested for "quantitative individual" and "aggregated" evaluations. A note on how the above metrics were computed is in [1].

(vi) The results of our experiments are shown in Figures 6 and 7, where each random graph is represented by a small circle. The x-coordinate corresponds to the metric value produced by OGDF, while the y-coordinate reflects the value obtained by DOMUS. Points below the bisector of the first quadrant (gray dashed line) indicate cases where DOMUS outperformed OGDF; points above indicate the opposite. To account for overlapping data points (caused by graphs with same metric values), each plot includes a red dashed trend line summarizing the comparative performance. We also report the coefficient of determination (R^2, with $0 \leq R^2 \leq 1$), with values closer to 1 indicating a better interpolation.



(a) Area.

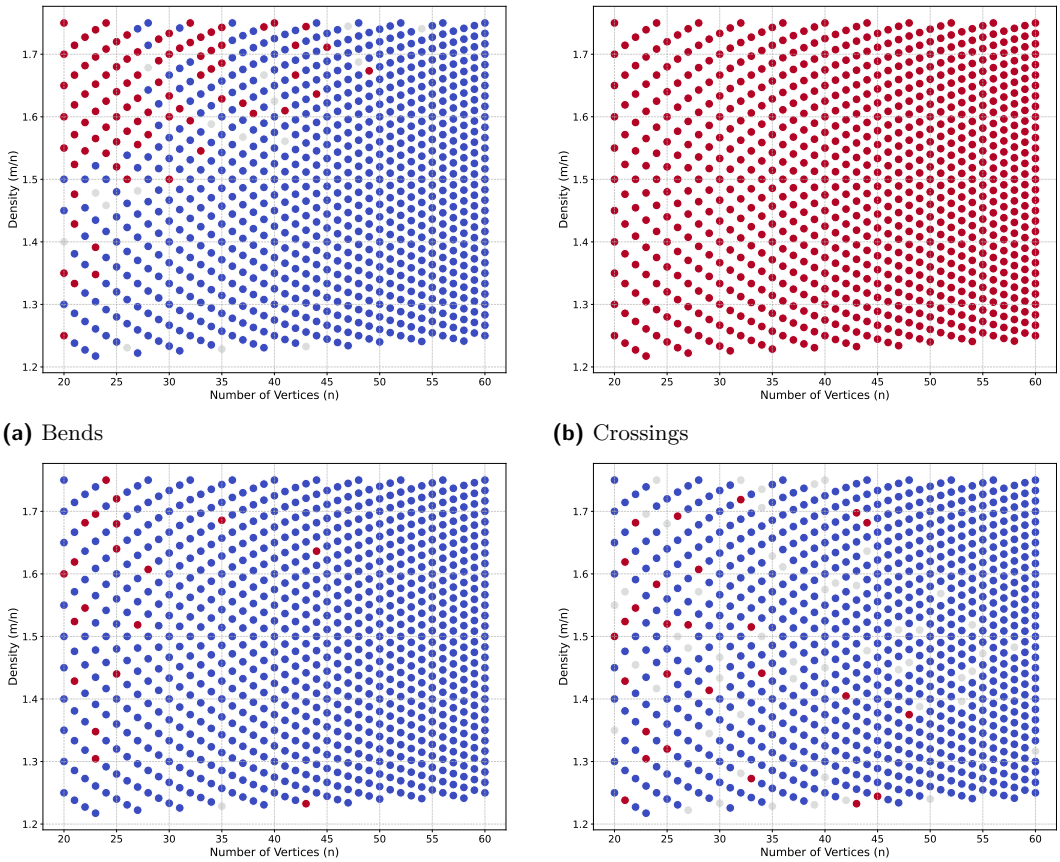
(b) Total Edge Length.

(c) Max Edge Length.

(d) Edge Length Deviation.

Figure 7 Effectiveness of OGDF and DOMUS "in vitro": Area and Edge Length.

Figure 6a focuses on Bends. DOMUS outperformed OGDF on 78% of the graphs and matched its performance on another 7%. The trend line indicates a modest quadratic improvement, a linear improvement of approximately 2%, and a constant improvement of about 2.5 bends per graph. Notably, 89 of the random graphs admit a rectilinear drawing. Conversely, as expected (Figure 6b), OGDF sharply outperforms DOMUS in terms of Crossings. In this case, 149 random graphs admit a planar drawing. Additional insights regarding bends are in Figures 6c and 6d, which report the Bends Deviation and the Max Bends per edge, respectively. For the former metric, DOMUS outperforms OGDF on 85% of the graphs. For the latter, DOMUS matches OGDF on 50% of the graphs and outperforms it in the remaining 50%. Since the possible values of the maximum number of bends per edge are relatively few, many graphs correspond to the same point. Because of this, in Figure 6d we represent with a variable size circle the number of graphs having the maximum number of bends per edge. The area of each circle is proportional to the corresponding number of graphs. These results indicate that our approach not only reduces the total number of bends but also achieves a more uniform distribution of bends across edges, thereby avoiding drawings with edges that contain long sequences of consecutive bends.



(a) Bends

(b) Crossings

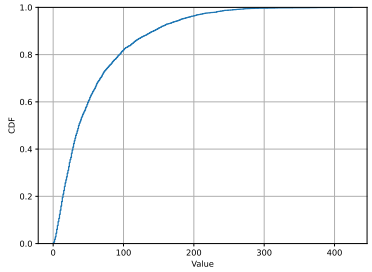
(c) Bends Deviation

(d) Max Bends

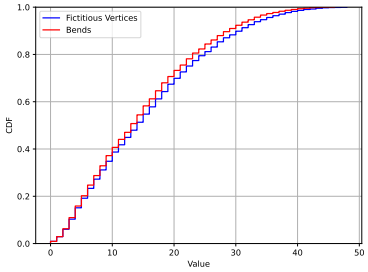
Figure 8 The effect of density: Bends and Crossings. Blue: DOMUS is better. Red: OGDF is better. Grey: parity.

DOMUS yields more compact drawings compared to those generated by OGDF (Figure 7a). Specifically, our approach results in smaller drawing areas for 78% of the graphs, and matches OGDF in 18% of the cases. The trend line indicates a consistent improvement, with an average area reduction of approximately 25%. We also achieve slightly better results than OGDF in terms of edge length. DOMUS produces drawings with lower Total Edge Length for 50% of the graphs and matches OGDF on 1% of them (Figure 7b). More significantly, DOMUS outperforms OGDF on Max Edge Length (Figure 7c) for 87% of the graphs and obtain equal results for an additional 10%. Similarly, for Edge Length Deviation (Figure 7d), DOMUS performs better on 89% of the graphs. These results indicate that SM not only tends to produce slightly shorter edges overall but also ensures a more uniform edge length distribution, thereby avoiding disproportionately long edges that can hinder readability and spatial coherence.

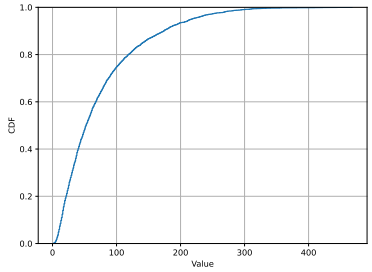
Figure 8 illustrates how the relative performance of DOMUS and OGDF varies with the number of vertices and the density of the input graphs. In each subfigure, a circle positioned at coordinates (x, y) represents all graphs with x vertices and density y. For a given metric M and each pair (x, y), let M_O (respectively, M_D) denote the average value of M for the drawings produced by OGDF (respectively, DOMUS) over all graphs with x vertices and density y. The circle at (x, y) is colored red, blue, or gray if $M_O < M_D$, $M_O > M_D$, or $M_O = M_D$,



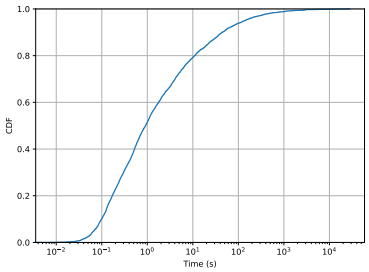
(a) Cycles added during the computation.



(b) Dummy vertices and bends.



(c) SAT invocations.



(d) Time.

Figure 9 Several aspects of the DOMUS computations.

respectively. Figure 8a shows that for small values of n, increasing graph density tends to favor OGDF. However, this advantage gradually diminishes as n increases. For the Area, we have a similar trend. The other metrics do not show any clear dependency on either the number of vertices or the density (Figures 8b–8d). [1] contains additional diagrams about the density.

(vii) To further characterize the computations performed by DOMUS, for each run on a single graph, we measured the following (see Figure 9).

First, we measured how many cycles were added to $\mathcal{C}$ during the computation. Each addition corresponds to a case where the current shape was found to be non-rectilinear drawable, triggering a new invocation of Shape Construction from Drawing Construction. In each of such invocations, the SAT solver was able to quickly find a satisfying assignment. The total number of Drawing Construction calls equals this number plus one. Figure 9a shows a cumulative distribution function (CDF) of these data. For instance, in 80% of the graphs, no more than 100 cycles were added to $\mathcal{C}$.

Second, we measured the number of dummy vertices introduced by the Shape Construction, corresponding to how often the Shape Construction called itself due to the absence of a shape satisfying the constraints. In each of these invocations, the formula given to the SAT solver was unsatisfiable. Figure 9b shows that at most 25 dummy vertices have been added for 80% of graphs. Such vertices do not necessarily become bends in the final drawing, since some of them can get an angle of π between their incident edges. So, a dummy vertex that does not become a bend can be viewed as a failure in the attempt to decrease the constraints of the drawing. Hence, we also computed how many dummy vertices actually became bends in the final drawing. This is an indirect measure of the effectiveness of using SAT unsatisfiability proofs to introduce bends. Figure 9b shows that almost all the dummy vertices became bends in the drawing.

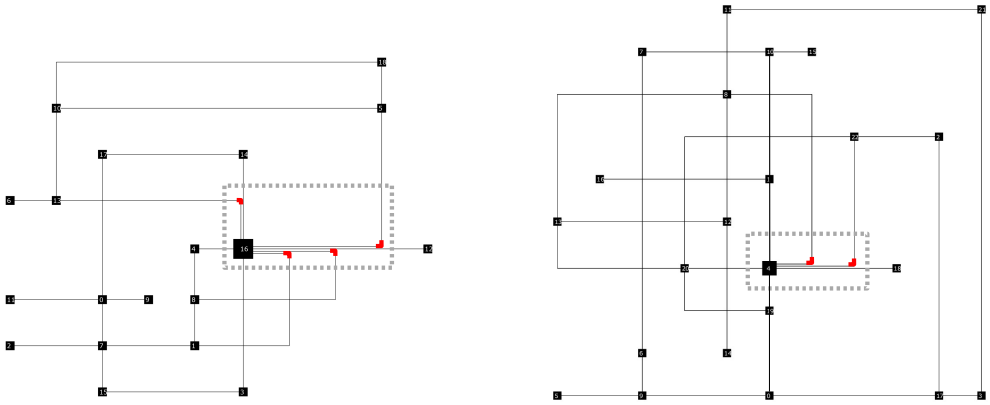


Figure 10 Illustrations of how DOMUS manages a vertex with degree greater than 4. The dashed boxes show the expansions and the red segments show the bends that are added to the drawing.

Third, we measured the total number of SAT solver invocations, computed as the sum of the Shape Construction self-invocations, the Shape Construction calls by the Drawing Construction, plus one. Figure 9c shows that at most 120 SAT invocations have been done for 80% of graphs. The similarity between Figure 9a and Figure 9c shows that most of the SAT solver calls (80%) depend on the need to add a cycle to $\mathcal{C}$. Also, the average number of Boolean variables and clauses in the input to the SAT solver were 352 and 1224, respectively.

Fourth, Figure 9d shows that computing a drawing for DOMUS required at most 10 seconds for 80% of graphs and that there are graphs that required more than 100 seconds, with an average latency of 70 seconds. OGDF did its work in an average of 0.06 seconds.

6 An Experimental Evaluation on the Rome Graphs

With small modifications, it is possible to apply our methodology to construct orthogonal drawings of graphs with degree greater than 4. In representing such graphs, we use the same convention introduced in [24]. Namely, the edges exiting the same side of a vertex can partially overlap (they are distanced by a very small amount) only for the first segment representing them. See Figure 10. The variation of the methodology works as follows. First, we modify the $\mathcal{F}_{G,\mathcal{C}}$ formula in such a way that if a vertex v has degree greater than 4 more than one edge can enter v from the same direction, while keeping the constraint that at least one edge enters v from all the directions. After that, all the machinery that looks for a shape, including the usage of the SAT solver, stays the same. Second, once a shape has been found, v is temporarily expanded into a box (which will not be shown in the drawing) and the edges incident to the same side of v are separated by introducing inside the box dummy vertices that become bends in the final drawing. After that, a metric is computed with the same technique as before. See drawings of Rome graphs in Figure 10.

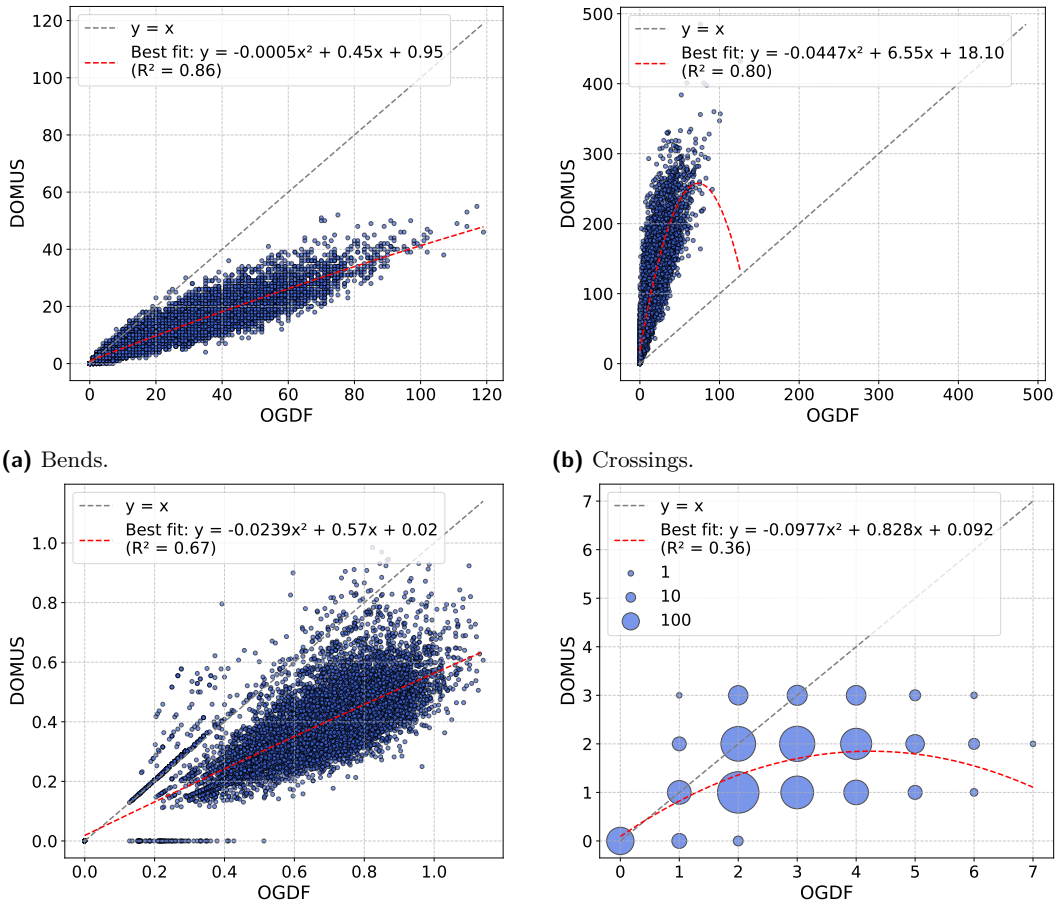
We compared OGDF and DOMUS in experiments that we call "in the wild" using the Rome Graphs dataset [20], which was originated by real-life applications and whose vertices have no degree restriction (see Figures 11 and 12). The experiments further increase the advantage of DOMUS with respect to OGDF (see Section 5) in terms of Bends (Figures 11a, 11c, and 11d) and confirm that OGDF performs much better in terms of Crossings (Figure 11b). This probably depends on the low densities of the Rome Graphs, which are close to planar in

terms of graph edit distance. Further, DOMUS performs better than OGDF in terms of Area, Max Edge Length, and Edge Length Deviation (Figures 12a, 12c, and 12d) and the two tools are comparable for Total Edge Length (Figure 12b). Hence, the presence of vertices of degree greater than 4 does not change too much the experimental results. OGDF was, even in this case, much faster than DOMUS, since the average duration of its computations was 0.07 seconds with a maximum of 3.3 seconds. The DOMUS computations took an average of 1.44 seconds, with a maximum of 769 seconds.

7 Conclusions and Open Problems

About forty years after the introduction of the Topology-Shape-Metrics (TSM) approach for orthogonal drawings, we propose a novel methodology that subverts the TSM pipeline by prioritizing bend minimization over crossing minimization. Our experimental results demonstrate that this shift yields improvements across most standard metrics used in Graph Drawing to evaluate the effectiveness of a layout algorithm.

As a final remark, we observe that our methodology can be exploited to construct drawings incorporating several types of constraints (for an introduction to constrained orthogonal drawings, see, e.g., [23, 41]). If the graph contains edges that should be drawn upward it is



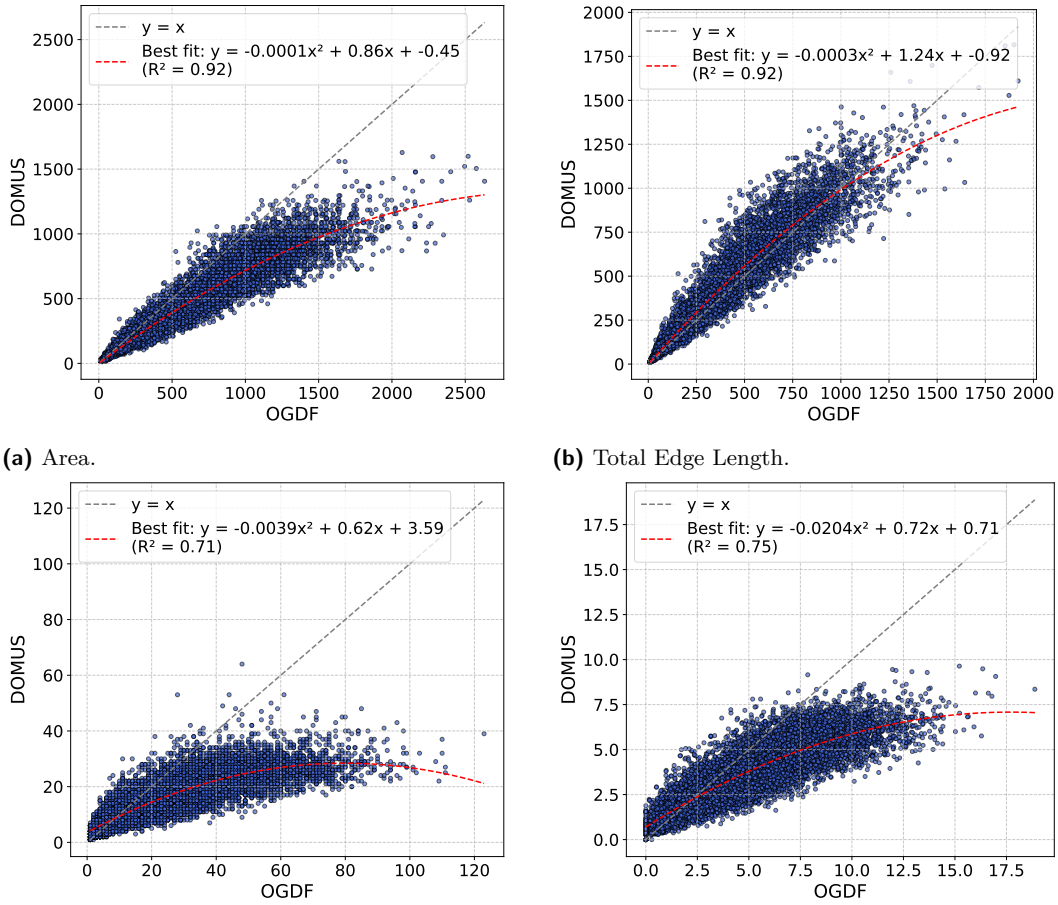
(a) Bends.

(b) Crossings.

(c) Bends Dev.

(d) Max Bends.

Figure 11 Effectiveness of OGDF and DOMUS "in the wild": Bends and Crossings.



(a) Area.

(b) Total Edge Length.

(c) Max Edge Length.

(d) Edge Length Deviation.

Figure 12 Effectiveness of OGDF and DOMUS "in the wild": Area and Edge Length.

easy to impose this by setting the value true for the corresponding u-variables of the $\mathcal{F}_{G,\mathcal{C}}$ formula. Similarly, giving the appropriate true-false values to the corresponding ℓ-, r-, d-, and u-variables it is possible to specify that certain edges should enter their end-vertices respecting some specific orientations (so called side-constraints). Further, it is easy to specify that a given edge should be horizontal or vertical or to impose the shape of entire sub-graphs.

We open several promising research directions, including the following:

- Does there exist, at least for some families of graphs, a "small" set of cycles whose completeness implies the completeness of all cycles in G? If so, can such a set be efficiently computed? In [1] it is shown that this problem is not trivial.
- Our methodology entirely disregards crossings. Is it possible to modify the approach to incorporate constraints that upper-bound the number of crossings?
- TSM and our methodology can be viewed as two extremes of a broader design space for constructing orthogonal drawings. Can hybrid methods be developed that combine elements of both approaches?
- In this paper, we employed compaction techniques originally developed within the TSM paradigm. It would be interesting to design compaction methods that do not work on the planarized underlying graph, but that only preserve the shape of the edges.

- Our approach can be leveraged to compute a maximal rectilinear drawable subgraph by iteratively invoking a SAT solver. Can the methodology be redesigned or optimized for greater efficiency when solving this specific problem?

For the sake of reproducibility all our software and our data sets are publicly available at `https://github.com/shape-metrics/domus`.

References

1 Giordano Andreola, Susanna Caroppo, Giuseppe Di Battista, Fabrizio Grosso, Maurizio Patrignani, and Allegra Strippoli. A walk on the wild side: a shape-first methodology for orthogonal drawings, 2025. arXiv:2508.19416.

2 Gilles Audemard and Laurent Simon. On the glucose SAT solver. *Int. J. Artif. Intell. Tools*, 27(1):1840001:1–1840001:25, 2018. doi:10.1142/S0218213018400018.

3 Sara Di Bartolomeo, Tarik Crnovrsanin, David Saffo, Eduardo Puerta, Connor Wilson, and Cody Dunne. Evaluating graph layout algorithms: A systematic review of methods and best practices. *Comput. Graph. Forum*, 43(6), 2024. doi:10.1111/CGF.15073.

4 Carlo Batini, Enrico Nardelli, and Roberto Tamassia. A layout algorithm for data flow diagrams. *IEEE Trans. Software Eng.*, 12(4):538–546, 1986. doi:10.1109/TSE.1986.6312901.

5 Carlo Batini, Maurizio Talamo, and Roberto Tamassia. Computer aided layout of entity relationship diagrams. *J. Syst. Softw.*, 4(2-3):163–173, 1984. doi:10.1016/0164-1212(84)90006-2.

6 Michael A. Bekos, Carla Binucci, Giuseppe Di Battista, Walter Didimo, Martin Gronemann, Karsten Klein, Maurizio Patrignani, and Ignaz Rutter. On turn-regular orthogonal representations. In David Auber and Pavel Valtr, editors, *Graph Drawing and Network Visualization - GD 2020*, volume 12590 of *LNCS*, pages 250–264. Springer, 2020. doi:10.1007/978-3-030-68766-3_20.

7 Michael A. Bekos, Carla Binucci, Giuseppe Di Battista, Walter Didimo, Martin Gronemann, Karsten Klein, Maurizio Patrignani, and Ignaz Rutter. On turn-regular orthogonal representations. *J. Graph Algorithms Appl.*, 26(3):285–306, 2022. doi:10.7155/JGAA.00595.

8 Michael A. Bekos, Michael Kaufmann, and Christian Zielke. The book embedding problem from a sat-solving perspective. In Emilio Di Giacomo and Anna Lubiw, editors, *Graph Drawing and Network Visualization - GD 2015*, volume 9411 of *LNCS*, pages 125–138. Springer, 2015. doi:10.1007/978-3-319-27261-0_11.

9 Therese Biedl, Thomas Bläsius, Benjamin Niedermann, Martin Nöllenburg, Roman Prutkin, and Ignaz Rutter. Using ILP/SAT to determine pathwidth, visibility representations, and other grid-based graph drawings. In Stephen K. Wismath and Alexander Wolff, editors, *Graph Drawing - 21st International Symposium, GD 2013*, volume 8242 of *LNCS*, pages 460–471. Springer, 2013. doi:10.1007/978-3-319-03841-4_40.

10 Therese Biedl and Goos Kant. A better heuristic for orthogonal graph drawings. In Jan van Leeuwen, editor, *Algorithms — ESA '94*, pages 24–35, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg. doi:10.1007/BFb0049394.

11 Therese C. Biedl and Goos Kant. A better heuristic for orthogonal graph drawings. *Comput. Geom.*, 9(3):159–180, 1998. doi:10.1016/S0925-7721(97)00026-6.

12 Timo Brand, Daniel Faber, Stephan Held, and Petra Mutzel. A customized sat-based solver for graph coloring. *CoRR*, abs/2504.04821, 2025. doi:10.48550/arXiv.2504.04821.

13 Markus Chimani, Carsten Gutwenger, Michael Jünger, Gunnar W Klau, Karsten Klein, Petra Mutzel, et al. The Open Graph Drawing Framework (OGDF). *Handbook of graph drawing and visualization*, 2011:543–569, 2013.

14 Markus Chimani and Robert Zeranski. Upward planarity testing via SAT. In Walter Didimo and Maurizio Patrignani, editors, *Graph Drawing - GD 2012*, volume 7704 of *LNCS*, pages 248–259. Springer, 2012. doi:10.1007/978-3-642-36763-2_22.

15 Markus Chimani and Robert Zeranski. Upward planarity testing: A computational study. In Stephen K. Wismath and Alexander Wolff, editors, *Graph Drawing - GD 2013*, volume 8242 of *LNCS*, pages 13–24. Springer, 2013. doi:10.1007/978-3-319-03841-4_2.

16 Markus Chimani and Robert Zeranski. Upward planarity testing in practice: SAT formulations and comparative study. *ACM J. Exp. Algorithmics*, 20:1.2:1.1–1.2:1.27, 2015. doi:10.1145/2699875.

17 Karl Däubel, Sven Jäger, Torsten Mütze, and Manfred Scheucher. On orthogonal symmetric chain decompositions. *Electron. J. Comb.*, 26(3):3, 2019. doi:10.37236/8531.

18 Giuseppe Di Battista and Walter Didimo. Gdtoolkit. In Roberto Tamassia, editor, *Handbook on Graph Drawing and Visualization*, pages 571–597. Chapman and Hall/CRC, 2013.

19 Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice-Hall, 1999.

20 Giuseppe Di Battista, Ashim Garg, Giuseppe Liotta, Roberto Tamassia, Emanuele Tassinari, and Francesco Vargiu. An experimental comparison of four graph drawing algorithms. *Comput. Geom.*, 7:303–325, 1997. doi:10.1016/S0925-7721(96)00005-3.

21 Peter Eades, Seok-Hee Hong, and Sheung-Hung Poon. On rectilinear drawing of graphs. In David Eppstein and Emden R. Gansner, editors, *Graph Drawing, GD 2009*, volume 5849 of *LNCS*, pages 232–243. Springer, 2009. doi:10.1007/978-3-642-11805-0_23.

22 Markus Eiglsperger. *Automatic layout of UML class diagrams: a topology-shape-metrics approach*. PhD thesis, University of Tübingen, Germany, 2003. URL: http://w210.ub.uni-tuebingen.de/dbt/volltexte/2004/1028/index.html.

23 Markus Eiglsperger, Ulrich Fößmeier, and Michael Kaufmann. Orthogonal graph drawing with constraints. In David B. Shmoys, editor, *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms, January 9-11, 2000, San Francisco, CA, USA*, pages 3–11. ACM/SIAM, 2000. URL: http://dl.acm.org/citation.cfm?id=338219.338225.

24 Ulrich Fößmeier and Michael Kaufmann. Drawing high degree graphs with low bend numbers. In Franz-Josef Brandenburg, editor, *Graph Drawing, Symposium on Graph Drawing, GD '95*, volume 1027 of *LNCS*, pages 254–266. Springer, 1995. doi:10.1007/BFB0021809.

25 Mohammad Ghoniem, Jean-Daniel Fekete, and Philippe Castagliola. On the readability of graphs using node-link and matrix-based representations: a controlled experiment and statistical analysis. *Inf. Vis.*, 4(2):114–135, 2005. doi:10.1057/PALGRAVE.IVS.9500092.

26 Tim Hegemann and Alexander Wolff. A simple pipeline for orthogonal graph drawing. In Michael A. Bekos and Markus Chimani, editors, *Graph Drawing and Network Visualization - GD 2023*, volume 14466 of *LNCS*, pages 170–186. Springer, 2023. doi:10.1007/978-3-031-49275-4_12.

27 Weidong Huang, Seok-Hee Hong, and Peter Eades. Effects of crossing angles. In *2008 IEEE Pacific Visualization Symposium*, pages 41–46, 2008. doi:10.1109/PACIFICVIS.2008.4475457.

28 Steve Kieffer, Tim Dwyer, Kim Marriott, and Michael Wybrow. Hola: Human-like orthogonal network layout. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):349–358, 2016. doi:10.1109/TVCG.2015.2467451.

29 Gunnar W. Klau and Petra Mutzel. Optimal compaction of orthogonal grid drawings. In Gérard Cornuéjols, Rainer E. Burkard, and Gerhard J. Woeginger, editors, *Integer Programming and Combinatorial Optimization, IPCO '99*, volume 1610 of *LNCS*, pages 304–319. Springer, 1999. doi:10.1007/3-540-48777-8_23.

30 Ján Manuch, Murray Patterson, Sheung-Hung Poon, and Chris Thachuk. Complexity of finding non-planar rectilinear drawings of graphs. In Ulrik Brandes and Sabine Cornelsen, editors, *Graph Drawing - GD 2010*, volume 6502 of *LNCS*, pages 305–316. Springer, 2010. doi:10.1007/978-3-642-18469-7_28.

31 Torsten Mütze and Manfred Scheucher. On l-shaped point set embeddings of trees: first non-embeddable examples. *J. Graph Algorithms Appl.*, 24(3):343–369, 2020. doi:10.7155/JGAA.00537.

32 Takao Nishizeki and Md. Saidur Rahman. *Planar Graph Drawing*, volume 12 of *Lecture Notes Series on Computing*. World Scientific, 2004. doi:10.1142/5648.

33 Achilleas Papakostas and Ioannis G. Tollis. Improved algorithms and bounds for orthogonal drawings. In Roberto Tamassia and Ioannis G. Tollis, editors, *Graph Drawing, DIMACS International Workshop, GD '94*, volume 894 of *LNCS*, pages 40–51. Springer, 1994. doi:10.1007/3-540-58950-3_355.

34 Achilleas Papakostas and Ioannis G. Tollis. Interactive orthogonal graph drawing. *IEEE Trans. Computers*, 47(11):1297–1309, 1998. doi:10.1109/12.736444.

35 Maurizio Patrignani. On the complexity of orthogonal compaction. *Computational Geometry: Theory and Applications*, 19(1):47–67, 2001. doi:10.1016/S0925-7721(01)00010-4.

36 Mauricio G.C. Resende and Celso C. Ribeiro. A grasp for graph planarization. *Networks: An International Journal*, 29(3):173–189, 1997. doi:10.1002/(SICI)1097-0037(199705)29:3\%3C173::AID-NET5\%3E3.0.CO;2-E.

37 Manfred Scheucher. Two disjoint 5-holes in point sets. *Comput. Geom.*, 91:101670, 2020. doi:10.1016/J.COMGEO.2020.101670.

38 Manfred Scheucher, Hendrik Schrezenmaier, and Raphael Steiner. A note on universal point sets for planar graphs. *CoRR*, abs/1811.06482, 2018. arXiv:1811.06482.

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

40 Roberto Tamassia. On embedding a graph in the grid with the minimum number of bends. *SIAM J. Comput.*, 16(3):421–444, 1987. doi:10.1137/0216030.

41 Roberto Tamassia. Constraints in graph drawing algorithms. *Constraints An Int. J.*, 3(1):87–120, 1998. doi:10.1023/A:1009760732249.

42 Roberto Tamassia, Carlo Batini, and Maurizio Talamo. An algorithm for automatic layout of entity-relationship diagrams. In Carl G. Davis, Sushil Jajodia, Peter A. Ng, and Raymond T. Yeh, editors, *Int. Conf. on Entity-Relationship Approach (ER'83)*, pages 421–439. North-Holland, 1983.

43 Roberto Tamassia, Giuseppe Di Battista, and Carlo Batini. Automatic graph drawing and readability of diagrams. *IEEE Trans. Syst. Man Cybern.*, 18(1):61–79, 1988. doi:10.1109/21.87055.

44 Gopalakrishnan Vijayan and Avi Wigderson. Rectilinear graphs and their embeddings. *SIAM Journal on Computing*, 14(2):355–372, 1985. doi:10.1137/0214027.