Lines: 440

On Compaction and Realizability of Almost Convex Octilinear Representations

Anonymous author(s)
Anonymous affiliation(s)

Abstract

Recently, it has been shown that Orthogonal Compaction admits an FPT algorithm with respect to the number of so-called *kitty corners*, which are specific pairs of reflex corners on the boundary of faces [Didimo et al., SOFSEM'23]. We investigate how this result extends to the *octilinear drawing model* and prove that Octilinear Compaction does not admit a PTAS even if all faces are convex. In contrast, we show that Octilinear Realizability is FPT in the number of reflex corners of interior faces. Finally, we show that the parameter cannot be relaxed by bounding the number of faces or the number of reflex corners per interior face. To do so, we prove that Octilinear Realizability remains NP-hard if (i) at most one face is not convex or (ii) if each interior face has at most eight reflex corners.

2012 ACM Subject Classification Human-centered computing → Graph drawings; Theory of computation → Fixed parameter tractability; Theory of computation → Approximation algorithms analysis

Keywords and phrases parameterized complexity, approximation, octilinear graph drawing

1 Introduction

Octilinear graph drawings have been the standard visualization paradigm [11, 14, 17] used in metro maps since the first map of the London Underground was designed by Henry Beck in 1933 [1]. In addition, they *extend* the popular *orthogonal graph drawing* style by two additional slopes (± 1) which can be useful in various diagramming applications such as UML or BPMN. Similar to studies on orthogonal graph drawings, research efforts on octilinear drawings have aimed to compute drawings with few bends per edge in small area [4, 5].

?

Due to the similarities to the orthogonal graph drawing model, one may be tempted to adopt techniques that have been successfully employed in the context of orthogonal graph drawing. However, many problems related to octilinear graph drawing turn out to be more difficult than the corresponding ones for the orthogonal model. For instance, it is NP-hard to compute an octilinear drawing with the fewest total number of bends if the embedding is fixed [15] while the corresponding problem for orthogonal drawings is polynomial time solvable [18].

In particular, we are interested in two constrained drawing problems where the input not only specifies the planar embedding but also the angles occurring between adjacent edges and the bends along each edge. In the Realizability problem one is then asked to compute any drawing satisfying the constraints whereas in the Compaction problem the goal is to find such a drawing using ~~fewest~~ the smallest area. While Orthogonal Realizability can be trivially solved [7, 18], Orthogonal Compaction is NP-hard [16] even if the graph is a cycle [10]. In contrast, already Octilinear Realizability is known to be NP-hard [3].

Recently, Didimo et al. have ~~it has been~~ shown that Orthogonal Compaction admits an FPT algorithm in terms of the number of so-called *kitty corners* [8] which are specific pairs of reflex corners occurring on the boundary of a face. Intuitively speaking, in the absence of such pairs of corners, each face can be partitioned into rectangular slices so that the width and the height of a drawing can be minimized independently [6]. For the FPT algorithm, Didimo et al. showed that it suffices to consider a number of such slicings that depends only on the number of

© Anonymous author(s);
licensed under Creative Commons License CC-BY 4.0
32nd International Symposium on Graph Drawing and Network Visualization (GD 2024).
Editors: Stefan Felsner and Karsten Klein; Article No. 23; pp. 23:1–23:15
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

kitty-corners. In this paper, we investigate to which extent bounding non-convexity may also facilitate Octilinear Realizability and Compaction.

Adjust capitalization

Our contribution. We investigate how bounding the number of reflex corners in an octilinear representation affects the computational complexity of Octilinear Compaction and Octilinear Realizability. We first show that Octilinear Compaction admits no $\frac{9}{4}$-approximation even if all faces are convex. In contrast, we then prove that Octilinear Realizability is polynomial~~-~~time solvable if all faces are convex, which also gives rise to an FPT algorithm parameterized by the number of reflex corners. On the other hand, we show that Octilinear Realizability is para-NP-hard[1] when parameterized by the ~~maximum~~ number of faces with reflex corners or by the maximum number of reflex corners per face.

Add references to sections.

2 Preliminaries

It is disputable whether this is part of the general notion of an octilinear drawing. I suggest to phrase it as an additional restriction that is investigated in this paper.

Formal Definitions. In an *octilinear drawing* of a *planar graph* $G = (V, E)$, each vertex $v \in V$ is represented by a point on the integer grid and each edge $e \in E$ is drawn as a sequence of horizontal, vertical and diagonal (with slope ± 1) line segments such that no two edge representations intersect except at common endpoints. Two important equivalence classes of octilinear graph drawings are *embeddings* and *octilinear representations*. Namely, an embedding contains all octilinear graph drawings with the same set of faces. An *octilinear representation*, on the other hand, contains all octilinear graph drawings with the same embedding that additionally have the same angles between consecutive edges around each vertex and the same sequence of bends along each edge.

A representation $\mathcal{R}$ can be alternatively defined via the set of constraints that a drawing must fulfill in order to belong to $\mathcal{R}$ (i.e., angles around each vertex and bends along each edge as defined above). Since the bends are predetermined, we can replace each bend by a dummy vertex of degree two. Hence, we assume in the following that no ~~each~~ edge has a ~~no~~ bend in $\mathcal{R}$. We call $\mathcal{R}$ *consistent* if the sum of rotations at interior angles in a counter-clockwise walk along the boundary of every internal face is 2π while the sum of such rotations along the boundary of the outer face is 6π. As rotations are measured with respect to a walk around the boundary of the face, there are both positive (convex angles) and negative rotations (reflex angles) as well as rotations with value zero (angles of π). Note that consistency is a necessary condition for $\mathcal{R} \neq \emptyset$. To this end, it is worth remarking that for the orthogonal model, it is also a sufficient condition. However, the following problem is known to be NP-hard [3]:

I'd write "the realizability of $\mathcal{R}$"

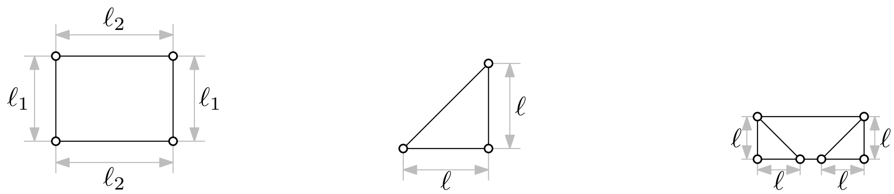
▶ **Problem 1** (Octilinear Realizability). *Given an octilinear representation $\mathcal{R}$, decide if $\mathcal{R} \neq \emptyset$.*

On the other hand, one may be interested to find a compact drawing for an octilinear representation $\mathcal{R}$ that is already known to be realizable:

▶ **Problem 2** (Octilinear Compaction). *Given an octilinear representation $\mathcal{R}$ that is realizable, i.e., $\mathcal{R} \neq \emptyset$, report a drawing $\Gamma \in \mathcal{R}$, such that the area of Γ is at most the area of Γ' for each $\Gamma' \in \mathcal{R}$.*

Almost Convex Representations. Orthogonal Compaction is in general NP-hard but can be efficiently solved under certain constraints [6, 13]. In the past few years, these concepts

[1] Para-NP-hardness means that the problem remains NP-hard for a bounded value of a parameter.



(a) **(b)** **(c)**

Figure 1 (a) Propagation gadget. (b) Rerouting gadget. (c) Copy gadget.

Gadgets used in the reduction from 3-SAT to Octilinear Compaction.

have resurfaced in the graph drawing community [2, 9] culminating in an FPT algorithm [8] parameterized by the number of so-called *kitty corners*. More precisely, kitty corners are *reflex corners* that "point" towards each other, ~~and~~ * a measure for non-convexity. We aim to apply the notion of limited non-convexity to investigate the parameterized complexity of Octilinear Realizability and Compaction. Our results concern a more relaxed parameter, namely, the number of reflex corners. We consider three variants of this parameter: (i) the *total number of reflex corners* ω in the entire representation, (ii) the ~~*maximum*~~ *number of faces* φ that contain at least one reflex corner, and (iii) the *maximum number of reflex corners per face* κ.

* their number is

ς ?

φ

κ

NP-hardness of Octilinear Compaction. We review the NP-hardness reduction from 3-SAT by Bekos et al. [3]. As a central concept, information is encoded in the length of specific edges of the drawing. *Propagation gadgets* each consist of a rectangular face, where the information can be propagated from either side to its opposite side; see Fig. 1a. Note that all sides of such a gadget may encode information. Similarly, *rerouting gadgets* are triangular faces consisting of horizontal, vertical and diagonal segments which allow to propagate from a vertical to a horizontal edge or vice-versa; see Fig. 1b. Moreover, two of these triangular faces can be combined with two straight-line edges to obtain a *copy gadget* as shown in Fig. 1c whose boundary necessarily contains four edges of equal length. These gadgets allow information to be propagated as required.

The variable u is not defined. Use ℓ_1 or simply u.

It remains to discuss how variables and clauses are encoded. The notion of a *unit edge length* ℓ_u is a crucial ingredient in the definition of the following gadgets. Namely, the *variable gadget* for a variable x consists of a single triangular shaped face as shown in Figs. 2a and 2b, so that the left side of the face is formed by three edges of length ℓ_u while the bottom side of the face consists of two edges. One of those is representing the literal x while the other edge represents the literal $\neg x$. As a result, it holds that $\ell(x) + \ell(\neg x) = 3\ell_u$.

Assume momentarily that $\ell_u = 1$. It is easy to see that $\ell(x), \ell(\neg x) \in \{1, 2\}$ on the integer grid. We assume that the literal whose corresponding edge has length 2 to be true and check with the clause gadget in Fig. 2c for clause $(a \vee b \vee c)$ whether at least one among the three literals a, b, and c is true, since the right side of the face has height more than three times the unit edge length $\ell_u = 1$.

However, it is possible to adjust the reduction to allow arbitrary values for ℓ_u. Namely, the *parity gadget* for variable x shown in Fig. 2e whose boundary is defined by edges of lengths ℓ_u, $\ell(x)$ and $\ell(\neg x)$ produces a crossing between the two *blocks* (highlighted blue and red in Fig. 2e) unless $\ell(x), \ell(\neg x) \in (0, 1.083\ell_u) \cup (1.917\ell_u, 3\ell_u)$; for a proof, see [3]. Observe that this small imprecision requires to adjust the clause gadget as well, see ~~as shown in~~ Fig. 2d. Namely, the sum of the edge lengths of the three literals must be at least $4\ell_u$, which can even be achieved if only one literal is true, as edges corresponding to false literals now can be

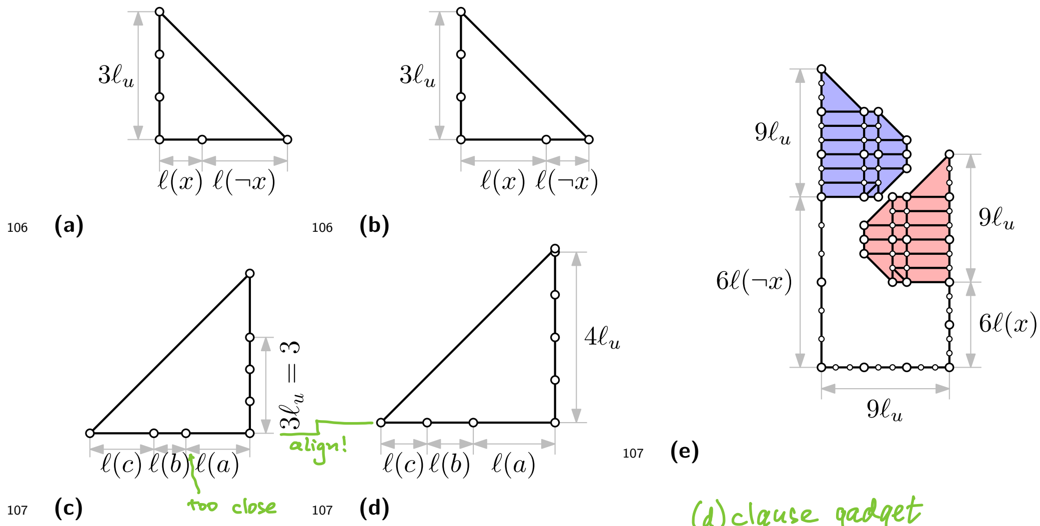


Figure 2 (a) and (b) Variable gadget for assignment $x = \perp$ and $x = \top$, resp. (c) ~~and (d)~~ Clause gadget where $\ell_u = 1$ with assignment $a = c = \top, b = \perp$ and (d) clause gadget where ℓ_u is arbitrary with assignment $a = \top, b = c = \perp$, ~~resp.~~ Observe that ~~in (d)~~ the two closely drawn vertices are connected by a vertical edge. (e) Parity gadget.

I suggest to use 'true' and 'false'.

slightly longer than ℓ_u. Also note that all reflex corners of the parity gadget occur on the top boundary of the outer face; see also Fig. 2e.

3 Approximability of Octilinear Compaction

In this section, we investigate the parameterized complexity of OCTILINEAR COMPACTION for $\omega = 0$, i.e., all faces are convex:

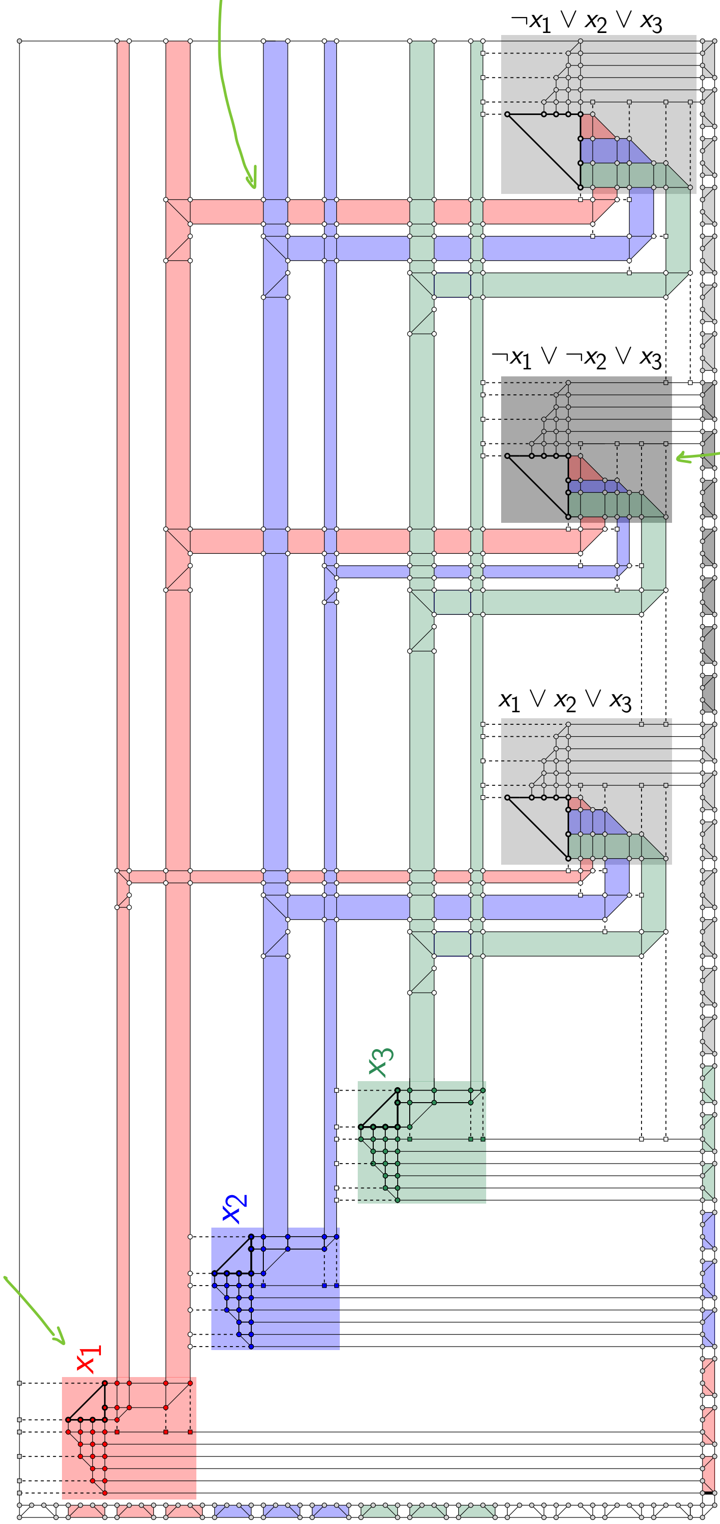
▶ **Theorem 1.** *Unless P=NP, there is no $\frac{9}{4}$-approximation for* OCTILINEAR COMPACTION *even if $\omega = 0$.*

Proof. We reduce from 3-SAT and closely follow the construction by Bekos et al. [3] with the variant where we assume that the unit length ℓ_u is equal to 1 as described in Section 2. This property is maintained by making the size of the output drawing dependent on the unit length ℓ_u. Namely, at the left boundary of the outer face, there exist $3n + 5$ copy gadgets while at the bottom boundary of the outer face, we have $3n + 7m$ copy gadgets where n and m are the number of variables and clauses of the 3-SAT formula, respectively; see Fig. 3.

More precisely, the first $3n$ copy gadgets at the bottom side connect to the n variable gadgets. The variable gadget of variable x_i appears above the variable gadget of variable x_{i+1}. The output literals of each variable are rerouted using rerouting gadgets so that the variable length is propagated towards the right via a series of propagation and copy gadgets, to which we refer as a *literal path* (colored in Fig. 3). Note that if $\ell_u = 1$, exactly one literal of each variable is true.

For each clause there are seven copy gadgets at the bottom side of the drawing and a clause gadget. In particular, the last three copies of the unit length of the associated copy gadgets are propagated via propagation and rerouting gadgets to the clause gadget. The clause gadget receives the information from the corresponding literal paths via the right copied length of a copy gadget on the literal path.

What about using transparency for the colored faces?



Why is this darker than the other two clauses?

Why rotated? Why different font?

Figure 3 Example of our reduction (Theorem 1) for the 3-SAT formula $(x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee x_3)$ and the satisfying assignment $x_1 = \bot$, $x_2 = x_3 = \top$.

Finally, we bound the upper and right side of the drawing by inserting a vertex at the top right corner. We connect this vertex with the topmost copy gadget horizontally and a path through the ends of each variable path and the rightmost copy gadget vertically. As a result, we will be able to establish area bounds based on the choice of the unit edge length ℓ_u. This will allow us to assume $\ell_u = 1$ for positive instances of 3-SAT and $\ell_u = 2$ for negative instances when minimizing the area. Thus, the reduction works without parity gadgets.

So far?

By now, there are some reflex corners in the drawing. However, we can add another horizontal or vertical segment by shooting a ray towards the face for which the vertex is reflex and subdividing the first segment crossed by the ray; see dashed segments and square-shaped vertices in Fig. 3. The obtained representation in fact has only convex faces, i.e., $\omega = 0$.

$l_u \to \ell_u$?

In any case, such similar variables should be avoided.

Since we inserted all the required gadgets for the NP-hardness reduction, we can conclude that there is no efficient algorithm computing a drawing with $\ell_u = 1$ for the obtained representation unless P=NP. It remains to discuss the area. The number of copies is chosen, so that all free edge lengths of the copy gadgets and the lengths of the edges connecting two copies of those gadgets at the top and bottom side can be chosen to be 1. Thus, each copy gadget is $2\ell_u + 1$ wide, in addition, between two copies there is an edge of length 1. Thus, depending on l_u, the width and height of the drawing are $h(l_u) = (2 \cdot l_u + 2) \cdot (3n + 5) + l_u$ and $w(l_u) = (2 \cdot l_u + 2) \cdot (3n + 7m) + l_u$, respectively. If a drawing with $\ell_u = 1$ exists, i.e., if the 3-SAT instance is satisfiable, the smallest drawing will have height and width $h(1) = 12n + 21$ and $w(1) = 12n + 28m + 1$, respectively. On the other hand, there is always a drawing with $\ell_u = 2$ as this allows to set $\ell(x) = \ell(\neg x) = 3$, which trivially fulfills the constraints of ~~each~~ every clause gadget. Hence, if the 3-SAT instance is unsatisfiable, we obtain a smallest drawing of width and height $h(2) = 18n + 32$ and $w(2) = 18n + 42m + 2$, respectively. Asssume now for a contradiction that there was a 9/4-approximation for Octilinear Compaction. Then, given a positive instance of 3-SAT, this approximation would yield a drawing Γ with area at most

$$\frac{9}{4}h(1) \cdot w(1) = \frac{3}{2}h(1) \cdot \frac{3}{2}w(1) = (18n + 31.5) \cdot (18n + 42m + 1.5) < h(2) \cdot w(2),$$

i.e., a drawing that is *smaller* than any drawing where $\ell_u = 2$. Thus, in Γ, we have that $\ell_u = 1$, i.e., we can actually decide in polynomial time whether there is a drawing with $l_u = 1$, a contradiction unless P=NP. ◀

▶ **Corollary 2.** *Unless P=NP, there is no PTAS for Octilinear Compaction.*

4 Octilinear Realizability of Almost Convex Representations

In contrast to Corollary 2, in this section we show that Octilinear Realizability can be efficiently solved if the input octilinear representation is almost convex. The main result of the section is that the problem is FPT with parameter ω. We first prove the following lemma, which ~~would be~~ is an interesting result by itself~~,~~ and that we use as ~~our~~ the main tool in our FPT algorithm. In the following, let $\mathcal{R}$ denote an octilinear representation of a graph G.

▶ **Lemma 3.** *Octilinear Realizability is in P when there is no reflex angle in any internal face and no inflex angle in the outer face in $\mathcal{R}$.*

nice!

defined?

convex?

Proof. If all faces are convex, a realizing drawing exists if and only if for each face the sum of widths of the edges forming the top boundary is equal to the sum of the widths of the edges forming the bottom boundary and the same holds for the left and the right boundary. We can model this behavior using two auxiliary flow networks similar to techniques used in

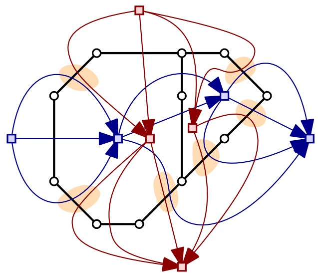


Figure 4 A representation $\mathcal{R}$ and auxiliary flow networks $G^{\rightarrow}$ (blue) and $G^{\downarrow}$ (red).

orthogonal graph drawing [7, 18]. Namely, $G^{\rightarrow}$ (blue graph in Fig. 4) contains a node for each internal face and two nodes s and t for the outer face. In addition, arc (f_1, f_2) exists if there is an edge e in $\mathcal{R}$ so that f_1 occurs to the left of e and f_2 to the right (for the outer face all outgoing and incoming edges are attached to s and t, resp.). It is easy to see that any valid flow from s to t with positive value along each arc describes an assignment of horizontal lengths to the edges of $\mathcal{R}$ satisfying the height constraint stated above. Similarly, we can define $G^{\downarrow}$ for the width constraint (red graph in Fig. 4). These two network flows can be easily described as a linear program.

To be more precise, we create a linear program formulation of the flow as follows. We associate each arc $a = (f_1, f_2) \in E(G^{\rightarrow}) \cup E(G^{\downarrow})$ with a variable x_a that must have strictly positive values, i.e., $x_a \geq 1$. Then, for each face f that is not the outer face, we create two constraints that ensure flow conservation in both $G^{\rightarrow}$ and $G^{\downarrow}$, namely,

$$\sum_{a=(f,f')\in E(G^{\rightarrow})} x_a = \sum_{a=(f',f)\in E(G^{\rightarrow})} x_a$$

and

$$\sum_{a=(f,f')\in E(G^{\downarrow})} x_a = \sum_{a=(f',f)\in E(G^{\downarrow})} x_a.$$

Note that diagonal segments create an arc in both $G^{\rightarrow}$ and $G^{\downarrow}$ (yellow highlights in Fig. 4). Thus, we must additionally require the flows along these two arcs to be the same as the height and the width of a diagonal at slope ± 1 are the same. These additional constraints are easily included in our linear program; for instance, we can simply use the same variable for both arcs.

The resulting linear program can be solved using polynomial time algorithms which yields a rational solution that can be encoded with a polynomial number of bits [12]. Hence, we can *scale* the solution to an integer solution requiring polynomial area which completes the proof. ◀

Given Lemma 3, our goal is now to extend it for fixed values of ω. In order to do that, we need further notation and intermediate results. In the following, let $\mathcal{R}$ denote an octilinear representation of a graph G. Recall that ω is ~~equal to~~ the number of reflex angles in $\mathcal{R}$. We first describe how to enrich G and $\mathcal{R}$ so that each face containing at least one reflex angle has size $O(\omega)$. These enrichments are denoted by the *shadow graph* and *representation* presented in Section 4.1. The key property of a shadow representation $\mathcal{R}'$ is that such a representation is realizable if and only if $\mathcal{R}$ is realizable (see Lemma 4). Hence, we can work on the shadow graph and on the shadow representation in order to obtain an FPT algorithm, which is described in Section 4.2.

4.1 Shadow graph and representation

We begin by defining the *shadow graph* G' and an octilinear representation $\mathcal{R}'$ of G', called the *shadow representation*, which are obtained from G and $\mathcal{R}$, respectively. Initially, we set $G' = G$ and $\mathcal{R}' = \mathcal{R}$. During the construction of the shadow graph and representation, we analyze each face f and enrich the graph considering the vertices incident to f and their angles inside f.

Consider first the internal faces. For each internal face f of G, consider each vertex v incident to f and add a vertex v' if the angle of v in f is either strictly convex or reflex. We say that v' is the *shadow vertex* of v. We order the shadow vertices following the order of the corresponding vertices and add a cycle connecting them. ~~that we denote by C_f~~. See the thicker red cycle in Figure 5b, where f is depicted in Figure 5a. Let C_f denote this cycle.

The newly created face f' bounded by C_f is called the *shadow face* of f. The angle inside f' of shadow vertex $v' \in C_f$ in G' is the same as the angle inside f of the vertex v in G. Finally, we use horizontal and vertical edges and, if necessary, subdivision vertices in order to connect C_f to the border of f in G'. We do this ~~that~~ as follow. Consider vertex v, its shadow vertex v', and the angle α at v in $\mathcal{R}$, which ~~that~~ is the same as the angle of v' in $\mathcal{R}'$. Consider the angle α_f at v' in the face between the boundary of f and C_f. Notice that either α is reflex and α_f is inflex or vice versa. We describe the construction for the case where α is reflex, the other case can be handled similarly. We proceed with the following case analysis, refer to Figure 5 where Figure 5a depicts f in G also showing the configurations of the angles described in $\mathcal{R}$, while Figure 5b shows the construction we are describing now.

$\alpha = \frac{7}{4}\pi$**.** For an example, see v_1 in Figure 5a. In this case v is incident to either a horizontal or a vertical edge on the border of f. Consider the first case, the second can be treated similarly. We consider the diagonal edge incident to v' and we subdivide it with a vertex s. Then we add an horizontal edge from v to this subdivision vertex, fixing the angles accordingly. See v_1' and s_1' in Figure 5b.

$\alpha = \frac{3}{2}\pi$**.** If v is incident to two diagonal edges (e.g., v_2 in Figure 5b), either its horizontal port or its vertical port is free inside f. Consider the first case (that is also the case of v_2 in Figure 5b). We add an edge connecting v to v' vertically (see v_2' in Figure 5b). Otherwise, we add a horizontal edge between v and v'. Next, suppose that v is incident to a horizontal and a vertical edge (e.g., v_3 in Figure 5a). In this case, we subdivide the horizontal edge of C_f incident to v' with a subdivision vertex s' and we add a vertical edge from v to s' (see v_3' and s_3' in Figure 5b). flip!

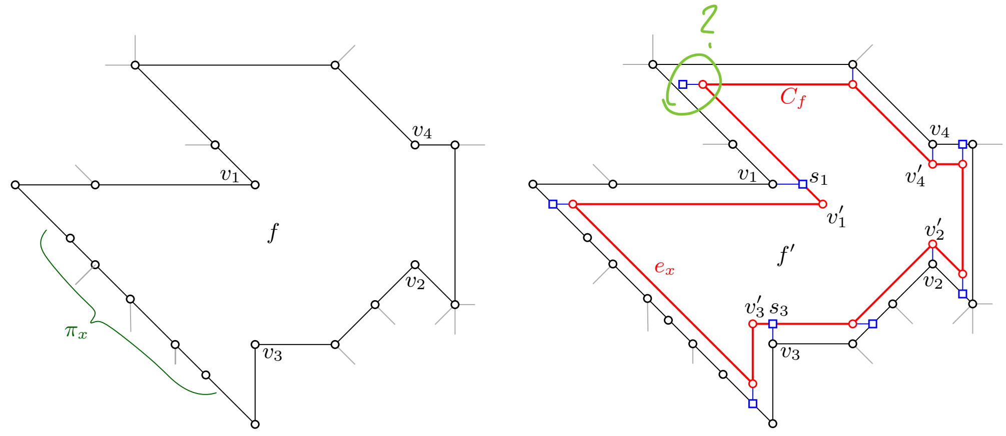
$\alpha = \frac{5}{4}\pi$**.** For an example, refer to v_4 in Figure 5a. We add an edge connecting v to v' that is vertical if v is incident to a horizontal edge or horizontal otherwise (see v_4' in Figure 5b).

$\alpha < \pi$**.** If α is strictly convex and α' is reflex, we do a symmetric operation. That is, we subdivide the facial cycle of f if necessary and select the orientation of the edge connecting v and v' based on the edges incident to the v'. See Figure 5 for examples.

f_out Consider now the outer face f_{out}. We construct $C_{f_{out}}$ similarly to how we constructed C_f for an internal face, inverting the construction as in this case $C_{f_{out}}$ is external to f_{out} while C_f was internal with respect to f. Refer to Figure 6. Figure 6a depicts an octilinear drawing of G that realizes a representation $\mathcal{R}$, while 6b depicts the corresponding octilinear drawing realizing the shadow representation $\mathcal{R}'$ of the shadow graph G'.

In the following, we denote by G' the shadow graph of G and by $\mathcal{R}'$ the shadow representation of G' induced by $\mathcal{R}$.

▶ **Lemma 4.** *$\mathcal{R}' \neq \emptyset$ if and only if $\mathcal{R} \neq \emptyset$.*



(a) G and $\mathcal{R}$ **(b)** G' and $\mathcal{R}'$

Move to page 8!

Figure 5 ~~Illustration for~~ the construction of G' and $\mathcal{R}'$.

Proof. Given an octilinear drawing of G' one can simply obtain an octilinear drawing of G by removing the shadow vertices and their incident edges. On the other hand, given an octilinear drawing of G one can simply obtain a drawing of G' by adding the vertices of C_f and the corresponding edges following the construction of $\mathcal{R}'$, eventually scaling the drawing to create space for such new vertices and edges when considering the internal faces. To this end, also note that the lengths of edges connecting vertices of f with vertices of C_f can be chosen arbitrarily small. → no if all vertices must lie on the integer grid, see line 51! ◀

An important property of the shadow graph is that every face of it containing a reflex angle does not have vertices forming π angles inside. We have the following lemma, that will let us define our FPT algorithm.

I guess you mean: The total complexity of the non-convex faces of G' is O(w)?

▶ **Lemma 5.** *All the faces of G' containing a reflex angle have size $O(\omega)$.*

Proof. Observe that an octilinear polygon with $O(\omega)$ reflex angles has $O(\omega)$ inflex angles. Concerning a face f of G with $O(\omega)$ reflex angles, it could have still $\Theta(n)$ vertices, since many of them could form angles of π inside f. The same does not hold by construction for the shadow faces in G'. By construction, the shadow faces of G' have $O(\omega)$ vertices as only strictly convex and reflex corners define their borders. All the $O(n)$ vertices forming an angle of π inside f correspond to an edge in f'. See, for example, the path π_x that is incident to f in Figure 5a and that corresponds to edge e_x incident to f' in ~~in~~ Figure 5b. By construction, all the faces of G incident to vertices that were forming angles of π inside f have no reflex angle inside. ◀

Maybe use Π to avoid confusion with π=180°.

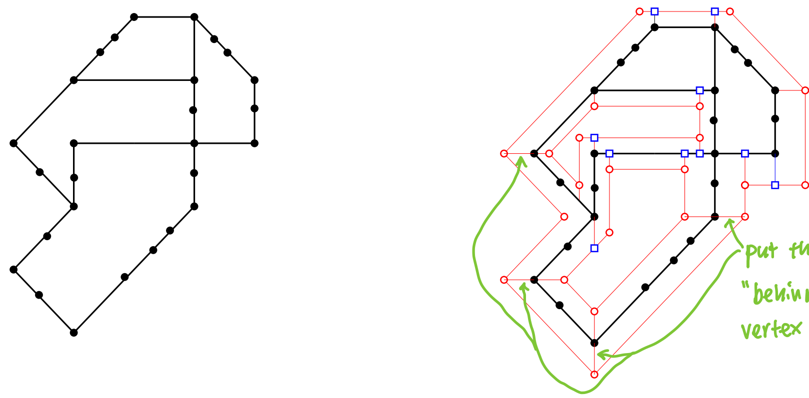
e_Π (why x? It's not defined.)

4.2 Planar Extensions and FPT Algorithm

Given G' and $\mathcal{R}'$, an *extension* of $\mathcal{R}'$ (and consequently ~~one~~ of G', which is the graph that $\mathcal{R}'$ represents) is defined as follow. We first add a 4-cycle J consisting of 4 vertices s_1, s_2, s_3, and s_4 in G'. ~~and~~ The angles around these vertices are $\frac{\pi}{2}$ inside the cycle and $\frac{3\pi}{2}$ outside. We denote by f_J the former. We define the outer face of G' to be equal to the face of this cycle having the reflex angles outside~~s~~. Hence, the rest of the graph has to be inside J. Let G'' be the plane graph obtained so far, which consists of G' and J. Let $\mathcal{R}''$ be the representation of G'' obtained as described above.

four

what?



put this red line segment "behind" the black and red vertex markers

(a) **(b)**

Figure 6 Illustration for the construction of the shadow graph and the shadow representation.

Consider G'' and $\mathcal{R}''$ and a shadow face f' of G'', whose border is C_f. Let v be a vertex of C_f forming a reflex angle in f'. Let w be another vertex of C_f. Let e be an edge of C_f or, if f is the outer face of G, it could be one of the four edges of J. Observe that in this case v is incident to the face in between the two disconnected components J and G' of G''. We define an operation that we ~~denote by~~ call *alignment*, that can take as input either v and w or v and e. The first operation of the alignment of v and w consists in adding an edge connecting v to w. The first operation of the alignment of v and e consists in subdividing e and adding an edge connecting v to the subdivision vertex d_v added in e. In both cases, the second and last operation of the alignment operation is to change the angles in the representation accordingly so that the added edge is horizontal. Figure 7 shows the octilinear representation of $\mathcal{R}'$ depicted in Figure 7 after several alignments of its vertices. We have that v is aligned to w and both v_1 and v_2 are aligned to (s_2, s_3), the respective dummy vertices are d_{v_1} and d_{v_2}.

Clarify whether the alignment "process" has been completed. (I think it has.)

We now define the pair $\langle G_+, \mathcal{R}_+ \rangle$, constructed starting from G'' and $\mathcal{R}''$ and iteratively choosing one possible alignment for a vertex v forming a reflex angle in a face of $\mathcal{R}''$. When two or more vertices align at a same edge (a, b), we chose an ordering of the subdivision vertices associated to such vertices when traversing such subdivided edge from a to b. See for example Figure 7, that shows the octilinear representation of an extension of $\mathcal{R}'$ depicted in Figure 6b. Notice that a single vertex can be considered at most twice by this procedure, as adding two horizontal edges to $\mathcal{R}''$ in any vertex makes sure that no angle around the vertex is larger than π. See for example vertex v in Figure 7.

Avoid double indices.

Observe that such ordering could imply that the obtained graph with the given rotation system at the vertices and the given outer face is not planar. For example, consider v_1 and v_2 that are both aligned to edge (s_2, s_3), in Figure 7. If the order of d_w and d_u was switched, i.e. if d_{v_1} was encoutered before d_{v_2} while traversing (s_2, s_3) from s_2 to s_3, the corresponding graph with the given rotation system and f_J as outer face ~~was not~~ planar.

d_{v_1} and d_{v_2}

would not be

In general, given one of the possible extensions $\langle G_+, \mathcal{R}_+ \rangle$, of G' and $\mathcal{R}'$ we have a graph that might be not planar given that an outer face and that the rotation system of the vertices

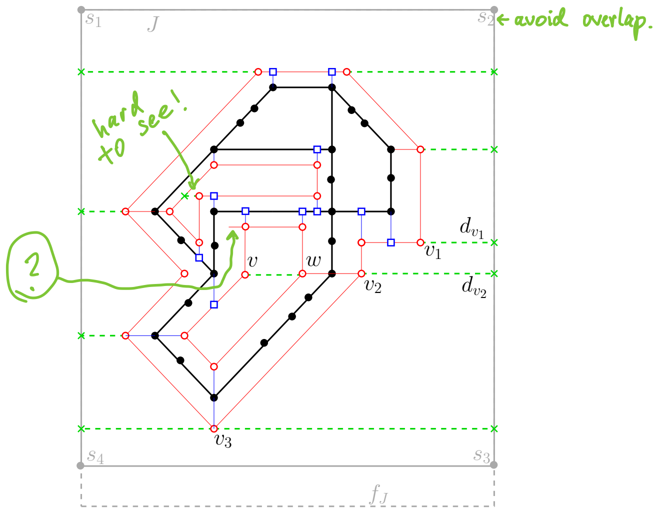


Figure 7 The octilinear representation of an extension of $\mathcal{R}'$ depicted in Figure 7.

in $\mathcal{R}'$ is fixed. We say that an extension is *planar* if the graph G'_+ with the rotation system defined by $\mathcal{R}'_+$ is planar and J as external cycle. ? The following two lemmas are the key ingredients for the proof of Theorem 8.

▶ **Lemma 6.** *There exists* $O\left(2^{\omega^2}\right)$ *possible extensions of* G' *and* $\mathcal{R}'$*.*

Proof. Every vertex forming a reflex angle can be alighed n ~~to~~ with an edge or a vertex incident to ~~a~~ the same face. By Lemma 5 it follows that every such vertex can be aligned to $O(\omega)$ elements, which ~~that~~ can be either vertices or edges. After choosing the alignments, we also have to order subdivision vertices that occur on the same edge. For each of the at most ω subdivided edges, there are at most ω subdivision vertices added. ~~i.e., which may~~ This give s rise to at most $\omega! = o\left(2^{\omega^2}\right)$ permutations. The lemma follows. ◀

▶ **Lemma 7.** $\mathcal{R}' \neq \emptyset \Leftrightarrow$ *There exists an extension* $\mathcal{R}'_+ \neq \emptyset$ *of* $\mathcal{R}'$*.*

Proof. The $\Rightarrow$ direction is immediate, as given a realization of $\mathcal{R}_+$ we can remove the edges added during the alignment operations, smooth the dummies that we added during the same operations, and remove J, and the rest of the drawing is a realization of $\mathcal{R}'$. For the other direction, Suppose that we have a realization $\Gamma \in \mathcal{R}'$. Every vertex at a reflex angle in an internal face is horizontally aligned with some edges incident to the same face. Concerning the outer face, if we add a drawing Γ_J of J so that Γ is in the internal face of Γ_J and such that all the internal angles are $\frac{\pi}{2}$, we have that every reflex angle in the external cycle of Γ is horizontally aligned either to another edge of the same cycle or to an edge of J. We can add such horizontal edges, eventually subdividing an edge edges ?, and then we obtain a realization of an extension of $\mathcal{R}'$. ◀

▶ **Theorem 8.** *Octlinear Realizability is FPT with respect to* ω*.*

Proof. Let $\mathcal{R}$ denote an octilinear representation of a graph G. If G has a vertex with degree at least 9 or if it is not planar or if $\mathcal{R}$ is not consistent, we reject the instance. Otherwise, we construct the shadow graph G' of G and the shadow representation $\mathcal{R}'$ of $\mathcal{R}$. Then we consider all the possible extensions of $\mathcal{R}'$ and we test the realizability of each extension using Lemma 3, which can be done as each extension has only four angles in the outer face, which are reflex angles, and no reflex angle inside by construction. We have that one of such extensions is realizable if and only if $\mathcal{R}$ is realizably by Lemmas ~~7 and Lemma 4~~ 4 and 7. We now discuss the ~~computational~~ running time of this procedure. In order to construct G' and $\mathcal{R}'$, it suffices to traverse each face of G $O(1)$ times and, for each vertex and each edge, to perform~~ing~~ $O(1)$ operations consisting in the case analysis depending on the angle at that vertex inside the face. Such operations are vertex addition, edge addition, and edge subdivision operations. Hence, constructing G' and $\mathcal{R}'$ can be done in polynomial time. The same holds for the construction of any extension of $\mathcal{R}'$. Hence, the computational time of the procedure is polynomial by Lemma 6 for a fixed value of parameter ω. ◀

This would be true even for an XP algorithm.

5 Para-NP-Hardness of Octilinear Realizability

recall meaning

Finally, we shift our attention ~~on~~ to the less restrictive parameters φ and κ and prove para-NP-hardness even for Octilinear Realizability:

▶ **Theorem 9.** *Octilinear Realizability remains NP-hard for $\varphi = 1$. In addition, it also remains NP-hard for $\kappa = 8$.*

Proof. We slightly modify the NP-hardness construction by Bekos et al. [3] as described in Section 2. More precisely, we start by putting a chain of copy gadgets on the bottom boundary of the drawing, so that each of the gadgets has two copies of the copied edge length at its top side; see Fig. 8. We will interpret the copied edge length as the unit length of the drawing. The length of this chain will be implicitly defined in the following discussion.

Figure

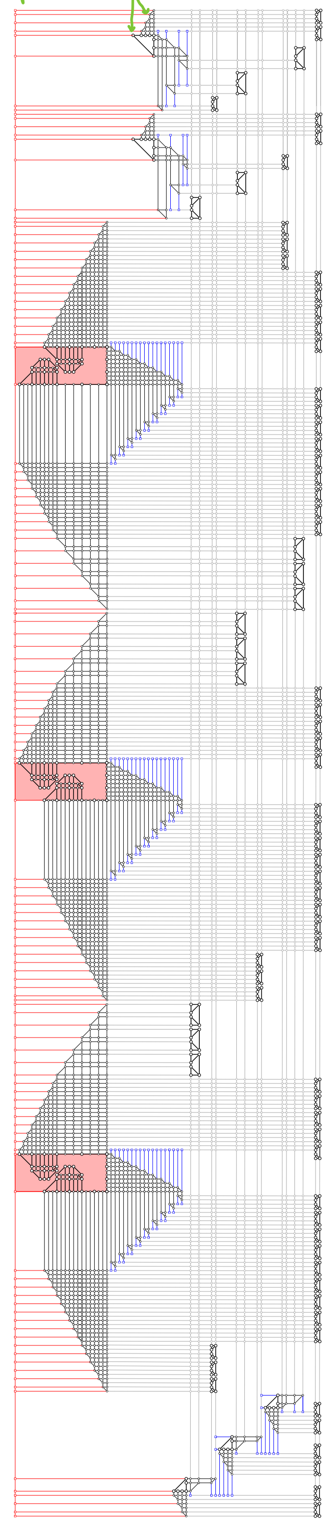
Then, we put the variable gadgets at the left side of the construction. Namely, we place the variable gadget for variable x_i above and to the left of the corresponding gadget for variable x_{i+1}; see Fig. 8. We connect each variable gadget to two copy gadgets on the bottom side of the drawing, that is, one copy remains unused. We then reroute the two literals of variable x_i so that a sequence of propagation and copy gadgets can propagate it rightwards to its usages within the parity and clause gadgets. Note that these *literal paths* of variable x_i occur above the variable gadget of variable x_{i+1}.

Next, we place the parity gadgets of all variables on the top side of the drawing, with the two blocks of the gadget being part of the outer face, above the variable gadget of x_1 so that the parity gadget of variable x_i appears to the left of the parity gadget of variable x_{i+1}. We connect each parity gadget to the two corresponding literal paths via three copy gadget on each of the paths. In addition, the unit length is obtained from fourteen copy gadgets that copy the unit length on the bottom side of the drawing; again one copy each remains unused; see Fig. 8.

Finally, it remains to position the clause gadgets to the right of the parity gadget of variable x_n. Again, we place the gadget for clause c_i to the left of the gadget for clause c_{i+1} so that the diagonal segment is on the outer face. Moreover, we connect the clause gadget of clause c_i to the three contained literal paths via a copy gadget on each of the paths (note that a copy remains unused). The unit length is provided from two copy gadgets on the bottom side of the drawing where one copy remains unused.

So far, we have described the black and gray parts of Fig. 8. Note that the discussion in Section 2 asserts that the obtained octilinear representation admits an octilinear drawing if and only if the encoded formula is satisfiable.

While the variable gadget of variable x_1, all parity gadgets and all clause gadgets will be located on the outer face, we have to insert additional segments to deal with reflex corners introduced by the variable gadgets for variables x_i with $i \geq 2$ and the rerouting gadgets to obtain $\varphi = 1$. We do so as follows; see the blue parts of Fig. 8. First, observe that each put the red line segments behind the vertex markers.



■ **Figure 8** Example of our reduction with formula $(x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee \neg x_3)$ and assignment $x_1 = x_2 = \top, x_3 = \bot$.

Both have 2 true and 1 false literal.
Maybe use $(\neg x_1 \vee \neg x_2 \vee \neg x_3)$ as second clause.

variable gadget for variable x_i (with $i \geq 2$) adds two reflex corners. We subdivide the first edge of the literal path of literal $\neg x_{i-1}$ above twice and connect the variable gadget's reflex corners to the created dummy vertices by vertical segments. In addition, for each reflex corner introduced by rerouting gadgets, we observe that there is a vertical edge directly to the left or to the right. We subdivide this *horizontally visible* edge and connect the reflex corner to the newly created dummy vertex by a horizontal segment. This completes the discussion for $\varphi = 1$.

It remains to augment our construction for $\varphi = 1$ to obtain the result for $\kappa = 8$; see the red parts of the drawing in Fig. 8. To do so, we add another dummy vertex above each reflex corner on the outer face except for those being part of a block of a parity gadget. We then connect each reflex corner with the corresponding dummy vertex with a vertical segment. Finally, we connect all dummy vertices with a horizontal path. It is straight-forward to see that the only remaining reflex corners occur in parity gadgets. Moreover, each parity gadget is part of a separate face; see the red shaded regions in Fig. 8. Thus, we conclude that each face contains at most $\kappa = 8$ reflex corners, which concludes the proof. ◀

▶ Remark 10. The reflex corners in parity gadgets cannot be eliminated with the methods presented in the proof of Theorem 9 as this would impede their functionality. Namely, one would be required to fix the order of the blocks inside such a gadget in order to add a segment incident to a dummy vertex subdividing an edge.

6 Open Problems

We conjecture that our results transfer to the smooth orthogonal drawing model where edges can be represented by straight-line segments, quarter circular arcs, half circular arcs and three-quarter circular arcs but must be attached to vertices in such a way that the tangent at the vertex is horizontal or vertical. In particular, to do so, one must define meaningful obstructions to convexity in this scenario. We remark that circular arc segments complicate the algorithmic question as our approach for convex octilinear representations in Theorem 3 cannot capture intersections between circular arcs.

Moreover, we are interested in investigating the complexity for different parameters. First, for some $\kappa < 8$, it may be the case that Octilinear Realizability becomes polynomial time solvable. In addition, our FPT algorithm may be parameterizable for a suitable definition of the kitty corners concept that is used for orthogonal graph drawing.

Finally, it remains open whether there is a constant factor approximation for Octilinear Compaction or if the problem is inapproximable.

References

1 Henry Beck. Tube map, *Waterlow & Sons Ltd., London*, 1933.

2 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. URL: https://doi.org/10.7155/jgaa.00595.

3 Michael A. Bekos, Henry Förster, and Michael Kaufmann. On smooth orthogonal and octilinear drawings: Relations, complexity and Kandinsky drawings. *Algorithmica*, 81(5):2046–2071, 2019. URL: https://doi.org/10.1007/s00453-018-0523-5.

4 Michael A. Bekos, Martin Gronemann, Michael Kaufmann, and Robert Krug. Planar octilinear drawings with one bend per edge. *J. Graph Algorithms Appl.*, 19(2):657–680, 2015. URL: https://doi.org/10.7155/jgaa.00369.

5 Michael A. Bekos, Michael Kaufmann, and Robert Krug. On the total number of bends for planar octilinear drawings. *J. Graph Algorithms Appl.*, 21(4):709–730, 2017. URL: `https://doi.org/10.7155/jgaa.00436`.

6 Stina S. Bridgeman, Giuseppe Di Battista, Walter Didimo, Giuseppe Liotta, Roberto Tamassia, and Luca Vismara. Turn-regularity and optimal area drawings of orthogonal representations. *Comput. Geom.*, 16(1):53–93, 2000. URL: `https://doi.org/10.1016/S0925-7721(99)00054-1`.

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

8 Walter Didimo, Siddharth Gupta, Philipp Kindermann, Giuseppe Liotta, Alexander Wolff, and Meirav Zehavi. Parameterized approaches to orthogonal compaction. In Leszek Gasieniec, editor, *SOFSEM 2023: Theory and Practice of Computer Science - 48th International Conference on Current Trends in Theory and Practice of Computer Science, SOFSEM 2023, ~~Nový Smokovec, Slovakia, January 15-18, 2023, Proceedings~~*, volume 13878 of *Lecture Notes in Computer Science*, pages 111–125. Springer, 2023. URL: `https://doi.org/10.1007/978-3-031-23101-8_8`.

9 Alexander M. Esser. Orthogonal compaction: Turn-regularity, complete extensions, and their common concept. In Ana Paula Cláudio, Kadi Bouatouch, Manuela Chessa, Alexis Paljic, Andreas Kerren, Christophe Hurter, Alain Trémeau, and Giovanni Maria Farinella, editors, *Computer Vision, Imaging and Computer Graphics Theory and Applications - 14th International Joint Conference, VISIGRAPP 2019, ~~Prague, Czech Republic, February 25-27, 2019, Revised Selected Papers~~*, volume 1182 of *Communications in Computer and Information Science*, pages 179–202. Springer, 2019. URL: `https://doi.org/10.1007/978-3-030-41590-7_8`.

10 William S. Evans, Krzysztof Fleszar, Philipp Kindermann, Noushin Saeedi, Chan-Su Shin, and Alexander Wolff. Minimum rectilinear polygons for given angle sequences. *Comput. Geom.*, 100:101820, 2022. URL: `https://doi.org/10.1016/j.comgeo.2021.101820`.

11 Seok-Hee Hong, Damian Merrick, and Hugo A. D. do Nascimento. Automatic visualisation of metro maps. *J. Vis. Lang. Comput.*, 17(3):203–224, 2006. URL: `https://doi.org/10.1016/j.jvlc.2005.09.001`.

12 Narendra Karmarkar. A new polynomial-time algorithm for linear programming. *Comb.*, 4(4):373–396, 1984. `doi:10.1007/BF02579150`.

13 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, 7th International IPCO Conference, Graz, Austria, June 9-11, 1999, Proceedings*, volume 1610 of *Lecture Notes in Computer Science*, pages 304–319. Springer, 1999. URL: `https://doi.org/10.1007/3-540-48777-8_23`.

14 Martin Nöllenburg and Alexander Wolff. Drawing and labeling high-quality metro maps by mixed-integer programming. *IEEE Trans. Vis. Comput. Graph.*, 17(5):626–641, 2011. URL: `https://doi.org/10.1109/TVCG.2010.81`.

15 Martin Nöllenburg. Automated drawings of metro maps, *Institut für Theoretische Informatik, Universität Karlsruhe (TH)*, Master's thesis, 2005. Add URL

16 Maurizio Patrignani. On the complexity of orthogonal compaction. *Comput. Geom.*, 19(1):47–67, 2001. URL: `https://doi.org/10.1016/S0925-7721(01)00010-4`.

17 Jonathan M. Stott, Peter Rodgers, Juan Carlos Martinez-Ovando, and Stephen G. Walker. Automatic metro map layout using multicriteria optimization. *IEEE Trans. Vis. Comput. Graph.*, 17(1):101–114, 2011. URL: `https://doi.org/10.1109/TVCG.2010.24`.

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