Orthogonal Graph Drawing with Inflexible Edges

Thomas Bläsius[✉], Sebastian Lehmann, and Ignaz Rutter

Faculty of Informatics, Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany
{thomas.blasius,sebastian.lehmann,ignaz.rutter}@kit.edu

Abstract. We consider the problem of creating plane orthogonal drawings of *4-planar graphs* (planar graphs with maximum degree 4) with constraints on the number of bends per edge. More precisely, we have a *flexibility function* assigning to each edge e a natural number $\text{flex}(e)$, its *flexibility*. The problem FlexDraw asks whether there exists an orthogonal drawing such that each edge e has at most $\text{flex}(e)$ bends. It is known that FlexDraw is NP-hard if $\text{flex}(e) = 0$ for every edge e [7]. On the other hand, FlexDraw can be solved efficiently if $\text{flex}(e) \geq 1$ [2] and is trivial if $\text{flex}(e) \geq 2$ [1] for every edge e.

To close the gap between the NP-hardness for $\text{flex}(e) = 0$ and the efficient algorithm for $\text{flex}(e) \geq 1$, we investigate the computational complexity of FlexDraw in case only few edges are *inflexible* (i.e., have flexibility 0). We show that for any $\varepsilon > 0$ FlexDraw is NP-complete for instances with $O(n^{\varepsilon})$ inflexible edges with pairwise distance $\Omega(n^{1-\varepsilon})$ (including the case where they induce a matching). On the other hand, we give an FPT-algorithm with running time $O(2^k \cdot n \cdot T_{\text{flow}}(n))$, where $T_{\text{flow}}(n)$ is the time necessary to compute a maximum flow in a planar flow network with multiple sources and sinks, and k is the number of inflexible edges having at least one endpoint of degree 4.

1 Introduction

Bend minimization in orthogonal drawings is a classical problem in the field of graph drawing. We consider the following problem called OptimalFlexDraw. The input is a 4-planar graph G (from now on all graphs are 4-planar) together with a cost function $\text{cost}_e \colon \mathbb{N} \to \mathbb{R} \cup \{\infty\}$ assigned to each edge. We want to find an orthogonal drawing Γ of G such that $\sum \text{cost}_e(\beta_e)$ is minimal, where β_e is the number of bends of e in Γ. The basic underlying decision problem FlexDraw restricts the cost function of every edge e to $\text{cost}_e(\beta) = 0$ for $\beta \in [0, \text{flex}(e)]$ and $\text{cost}_e(\beta) = \infty$ otherwise, and asks whether there exists a *valid* drawing (i.e., a drawing with finite cost). The value $\text{flex}(e)$ is called the *flexibility* of e. Edges with flexibility 0 are called *inflexible*. Note that FlexDraw represents the important base case of testing for the existence of a drawing with cost 0 that is included in solving OptimalFlexDraw.

Garg and Tamassia [7] show that FlexDraw is NP-hard in this generality, by showing that it is NP-hard if every edge is inflexible. For special cases, namely

Partially supported by grant WA 654/21-1 of the German Research Foundation (DFG).

V.Th. Paschos and P. Widmayer (Eds.): CIAC 2015, LNCS 9079, pp. 61–73, 2015.
DOI: 10.1007/978-3-319-18173-8_4

planar graphs with maximum degree 3 and series-parallel graphs, Di Battista et al. [5] give an algorithm minimizing the total number of bends, which solves OptimalFlexDraw with $\text{cost}_e(\beta) = \beta$ for each edge e. Their approach can be used to solve FlexDraw, as edges with higher flexibility can be modeled by a path of inflexible edges. Biedl and Kant [1] show that every 4-planar graph (except for the octahedron) admits an orthogonal drawing with at most two bends per edge. Thus, FlexDraw is trivial if the flexibility of every edge is at least 2. Bläsius et al. [2,3] tackle the NP-hard problems FlexDraw and OptimalFlexDraw by not counting the first bend on every edge. They give a polynomial time algorithm solving FlexDraw if the flexibility of every edge is at least 1 [2]. Moreover, they show how to efficiently solve OptimalFlexDraw if the cost function of every edge is convex and allows the first bend for free [3].

When restricting the allowed drawings to those with a specific planar embedding, the problem OptimalFlexDraw becomes significantly easier. Tamassia [9] shows how to find a drawing with as few bends as possible by computing a flow in a planar flow network. This flow network directly extends to a solution of OptimalFlexDraw with fixed planar embedding, if all cost functions are convex. Cornelsen and Karrenbauer [4] recently showed, that this kind of flow network can be solved in $O(n^{3/2})$ time.

Contribution and Outline. In this work we consider OptimalFlexDraw for instances that may contain inflexible edges, closing the gap between the general NP-hardness result [7] and the polynomial-time algorithms in the absence of inflexible edges [2,3]. After presenting some preliminaries in Section 2, we show in Section 3 that FlexDraw remains NP-hard even for instances with only $O(n^\varepsilon)$ (for any $\varepsilon > 0$) inflexible edges that are distributed evenly over the graph, i.e., they have pairwise distance $\Omega(n^{1-\varepsilon})$. This includes the cases where the inflexible edges are restricted to form very simple structures such as a matching.

On the positive side, we describe a general algorithm that can be used to solve OptimalFlexDraw by solving smaller subproblems (Section 4). This provides a framework for the unified description of bend minimization algorithms which covers both, previous work and results presented in this paper. We use this framework in Section 5 to solve OptimalFlexDraw for series-parallel graphs with monotone cost functions. This extends the algorithm by Di Battista et al. [5] to non-biconnected series-parallel graphs and thus solves one of their open problems. Moreover, we allow a significantly larger set of cost functions (in particular, the cost functions may be non-convex).

In Section 6, we present our main result, which is an FPT-algorithm with running time $O(2^k \cdot n \cdot T_{\text{flow}}(n))$, where k is the number of inflexible edges incident to degree-4 vertices, and $T_{\text{flow}}(n)$ is the time necessary to compute a maximum flow in a planar flow network of size n with multiple sources and sinks. Note that we can allow an arbitrary number of edges whose endpoints both have degree at most 3 to be inflexible without increasing the running time. Thus, our algorithm can also test the existence of a 0-bend drawing (all edges are inflexible) in FPT-time with respect to the number of degree-4 nodes. This partially solves another open problem of Di Battista et al. [5]. We conclude with open questions in Section 7.

Due to space constraints, we omit or only sketch several proofs. A full version with detailed proofs is available [3].

2 Preliminaries

Connectivity and the Composition of Graphs. A graph G is *connected* if there exists a path between every pair of vertices. A *separating k-set S* is a subset of vertices of G such that $G - S$ is not connected. Separating 1-sets are called *cutvertices* and separating 2-sets *separation pairs*. A connected graph without cutvertices is *biconnected* and a biconnected graph without separation pairs is *triconnected*. The *blocks* of a connected graph are its maximal (with respect to inclusion) biconnected subgraphs.

An *st-graph G* is a graph with two designated vertices s and t, its *poles*, such that $G + st$ is biconnected and planar. Let G_1 and G_2 be two st-graphs with poles s_1, t_1 and s_2, t_2, respectively. The *series composition* G of G_1 and G_2 is the union of G_1 and G_2 where t_1 is identified with s_2. Clearly, G is again an st-graph with the poles s_1 and t_2. In the *parallel composition* G of G_1 and G_2 the vertices s_1 and s_2 and the vertices t_1 and t_2 are identified with each other and form the poles of G. An st-graph is *series-parallel*, if it is a single edge or the series or parallel composition of two series-parallel graphs.

To be able to compose all st-graphs, we need a third composition. Let $G_1, \dots, G_\ell$ be a set of st-graphs with poles s_i and t_i associated with G_i. Let H be an st-graph with poles s and t such that $H + st$ is triconnected and let $e_1, \dots, e_\ell$ be the edges of H. The *rigid composition* G with respect to the so-called *skeleton* H is obtained by replacing each edge e_i of H by the graph G_i, identifying the endpoints of e_i with the poles of G_i. It follows from the theory of SPQR-trees that every st-graph is either a single edge or the series, parallel or rigid composition of st-graphs [6].

Orthogonal Representation. To handle orthogonal drawings of a graph G, we use the abstract concept of orthogonal representations neglecting distances in a drawing. Orthogonal representations were introduced by Tamassia [9], however, we use a slight modification that makes it easier to work with, as bends of edges and bends at vertices are handled the same. Let Γ be a *normalized* orthogonal drawing of G, i.e., every edge has only bends in one direction. If additional bends cannot improve the drawing (i.e., costs are monotonically increasing), a normalized optimal drawing exists [9]. All orthogonal drawings we consider are normalized. We assume that G is biconnected. This simplifies the description, as each edge and vertex has at most one incidence to a face. All definitions extend to connected graphs.

Let e be an edge in G that has β bends in Γ and let f be a face incident to e. We define the *rotation* of e in f as $\operatorname{rot}(e_f) = \beta$ and $\operatorname{rot}(e_f) = -\beta$ if the bends of e form 90° and 270° angles in f, respectively. For a vertex v forming the angle

α in the face f, we define $\text{rot}(v_f) = 2 - \alpha/90°$. Note that, when traversing a face of G in clockwise (counter-clockwise for the outer face) direction, the right and left bends correspond to rotations of 1 and -1, respectively (we may have two left bends at once at vertices of degree 1). The values for the rotations we obtain from a drawing Γ satisfy the following properties; see Fig. 1a.

(1) The sum over all rotations in a face is 4 (-4 for the outer face).
(2) For every edge e with incident faces f_ℓ and f_r we have $\text{rot}(e_{f_\ell})+\text{rot}(e_{f_r}) = 0$.
(3) The sum of rotations around a vertex v is $2 \cdot \deg(v) - 4$.
(4) The rotations at vertices lie in the range $[-2, 1]$.

Let $\mathcal{R}$ be a structure consisting of an embedding of G plus a set of values fixing the rotation for every vertex-face and edge-face incidence. We call $\mathcal{R}$ an *orthogonal representation* of G if the rotation values satisfy the above properties (1)–(4). Given an orthogonal representation $\mathcal{R}$, a drawing inducing the specified rotation values exists and can be computed efficiently [9].

Orthogonal Representations and Bends of st-Graphs. We extend the notion of rotation to paths; conceptually this is very similar to spirality [5]. Let π be a path from vertex u to vertex v. We define the rotation of π (denoted by $\text{rot}(\pi)$) to be the number of bends to the right minus the number of bends to the left when traversing π from u to v.

There are two special paths in an st-graph G. Let s and t be the poles of G and let $\mathcal{R}$ be an orthogonal representation with s and t on the outer face. Then $\pi(s,t)$ denotes the path from s to t when traversing the outer face of G in counter-clockwise direction. Similarly, $\pi(t,s)$ is the path from t to s. We define the *number of bends* of $\mathcal{R}$ to be $\max\{|\text{rot}(\pi(s,t))|, |\text{rot}(\pi(t,s))|\}$. Note that a single edge $e = st$ is also an st-graph. Note further that the notions of the number of bends of the edge e and the number of bends of the st-graph e coincide. Thus, the above definition is consistent.

When considering orthogonal representations of st-graphs, we always require the poles s and t to be on the outer face. We say that the vertex s has σ *occupied incidences* if $\text{rot}(s_f) = \sigma - 3$ where f is the outer face. We also say that s has $4-\sigma$ *free incidences* in the outer face. If the poles s and t have σ and τ occupied incidences in $\mathcal{R}$, respectively, we say that $\mathcal{R}$ is a (σ,τ)*-orthogonal representation*; see Fig. 1b.

Note that $\text{rot}(\pi(s,t))$ and $\text{rot}(\pi(t,s))$ together with the number of occupied incidences σ and τ basically describe the outer shape of G and thus how it has to be treated if it is a subgraph of some larger graph. Using the bends of $\mathcal{R}$ instead of the rotations of $\pi(s,t)$ and $\pi(t,s)$ implicitly allows to mirror the orthogonal representation (and thus exchanging $\pi(s,t)$ and $\pi(t,s)$).

Thick Edges. In the basic formulation of an orthogonal representation, every edge *occupies* exactly one incidence at each of its endpoints. We introduce *thick edges* that may occupy more than one incidence at each endpoint to represent larger subgraphs. Let $e = st$ be an edge in G. We say that e is a (σ,τ)*-edge* if e is defined to occupy σ and τ incidences at s and t, respectively. Note that the total

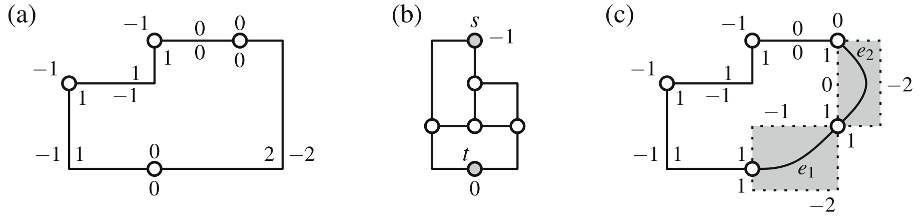

Fig. 1. (a) An orthogonal drawing together with its orthogonal representation given by the rotation values. (b) A $(2,3)$-orthogonal representation (s and t have 2 and 1 free incidences, respectively). (c) An orthogonal representation with thick edges e_1 and e_2. The gray boxes indicate how many attachments the thick edges occupy, i.e., e_1 is a $(2,3)$-edge and e_2 is a $(2,2)$-edge. Both thick edges have two bends.

amount of occupied incidences of a vertex in G must not exceed 4. With this extended notion of edges, we define a structure $\mathcal{R}$ consisting of an embedding of G plus a set of values for all rotations to be an *orthogonal representation* if it satisfies the following (slightly extended) properties; see Fig. 1c.

(1) The sum over all rotations in a face is 4 (-4 for the outer face).
(2) For every (σ,τ)-edge e with incident faces f_ℓ and f_r we have $\operatorname{rot}(e_{f_\ell}) + \operatorname{rot}(e_{f_r}) = 2 - (\sigma + \tau)$.
(3) The sum of rotations around a vertex v with incident edges $e_1, \dots, e_\ell$ occupying $\sigma_1, \dots, \sigma_\ell$ incidences of v is $\sum(\sigma_i + 1) - 4$
(4) The rotations at vertices lie in the range $[-2, 1]$.

Note that requiring every edge to be a $(1,1)$-edge in this definition of an orthogonal representation exactly yields the previous definition without thick edges. The *number of bends* of a (thick) edge e incident to the faces f_ℓ and f_r is $\max\{|\operatorname{rot}(e_{f_\ell})|, |\operatorname{rot}(e_{f_r})|\}$. Note that this is again consistent with the definition of number of bends of st-graphs and normal edges. Unsurprisingly, replacing a (σ,τ)-edge with β bends in an orthogonal representation by a (σ,τ)-orthogonal representation with β bends of an arbitrary st-graph yields a valid orthogonal representation [2, Lemma 5].

3 A Matching of Inflexible Edges

In this section, we show that FlexDraw is NP-complete even if the inflexible edges form a matching. In fact, we show the stronger result of NP-hardness of instances with $O(n^\varepsilon)$ inflexible edges (for $\varepsilon > 0$) even if these edges are distributed evenly over the graph, i.e., they have pairwise distance $\Omega(n^{1-\varepsilon})$.

We adapt the proof of NP-hardness by Garg and Tamassia [7] for the case that all edges of an instance of FlexDraw are inflexible. For a given instance of Nae-3Sat (Not All Equal 3SAT) they show how to construct a graph G that admits an orthogonal representation without bends if and only if the instance of Nae-3Sat is satisfiable. The graph G is obtained by first constructing a graph F that has a unique planar embedding [7, Lemma 5.1] and replacing the edges of F by larger st-graphs. These graphs have degree-1 poles and their embedding is fixed up to a flip, which implies the following lemma.

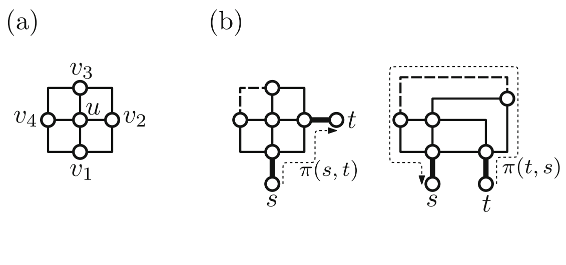

Fig. 2. The bold edges are inflexible; dashed edges have flexibility 2; all other edges have flexibility 1. (a) The wheel W_4. (b) The bend gadget $B_{1,2}$. (c) The gadget W'_3 for replacing degree-3 vertices. The marked subgraphs are bend gadgets.

Lemma 1 (Garg and Tamassia [7]). FlexDraw *is NP-hard, even if the order of edges around each vertex is fixed up to reversal.*

We assume that our instances do not contain degree-2 vertices; their incident edges can be replaced by a single edge with higher flexibility. In the following, we first show how to replace vertices of degree 3 by graphs of constant size such that each inflexible edge is incident to two vertices of degree 4. Afterwards, we can replace degree-4 vertices by smaller subgraphs with positive flexibility, which increases the distance between the inflexible edges.

The shape of the outer face of the wheel W_4 (with flexibility 1 on each edge) is the same in every valid orthogonal representation [2]; see Fig. 2a. It follows directly that the st-graph in Fig. 2b, which we call *bend gadget* and denote by $B_{1,2}$, has either one or two bends in each orthogonal representation. This ensures that replacing a degree-3 vertex by the construction W'_3 (which is basically an enhanced wheel of size 3) shown in Fig. 2c results in an equivalent instance of FlexDraw. Note that, after replacing each degree-3 vertex by a copy of W'_3, all endpoints of inflexible edges have degree 4. We obtain the following lemma.

Lemma 2. FlexDraw *is NP-hard, even if the endpoints of each inflexible edge have degree 4 and if the order of edges around each vertex is fixed up to reversal.*

Similarly, replacing the vertices incident to inflexible edges by copies of W_4 (whose edges have flexibility 1), increases the distance between any pair of inflexible edges by at least 1. Note that this does not increase the number of inflexible edges. Iterative replacement yields the following theorem.

Theorem 1. FlexDraw *is NP-complete even for instances of size n with $O(n^{\varepsilon})$ inflexible edges with pairwise distance $\Omega(n^{1-\varepsilon})$.*

The instances described above may contain edges with flexibility larger than 1. By iteratively replacing an edge e with flexibility $\text{flex}(e) > 1$ by a chain consisting of an edge with flexibility 1, the wheel W_4, and an edge with flexibility $\text{flex}(e) - 1$, we obtain an equivalent instance where all edges have flexibility 1 or 0.

4 The General Algorithm

In this section we describe a general algorithm that can be used to solve OptimalFlexDraw by solving smaller subproblems for the different types of graph compositions. To this end, we start with the definition of cost functions for subgraphs. The *cost function* $\text{cost}(\cdot)$ of an st-graph G is defined such that $\text{cost}(\beta)$ is the minimum cost of all orthogonal representations of G with β bends. The *(σ, τ)-cost function* $\text{cost}_\tau^\sigma(\cdot)$ of G is defined analogously by setting $\text{cost}_\tau^\sigma(\beta)$ to the minimum cost of all (σ, τ)-orthogonal representations of G with β bends. Clearly, $\sigma, \tau \in \{1, \dots 4\}$, though, for a fixed graph G, not all values may be possible. If for example $\deg(s) = 1$, then σ is 1 for every orthogonal representation of G. Note that there is a lower bound on the number of bends depending on σ and τ. For example, a $(2, 2)$-orthogonal representation has at least one bend and thus $\text{cost}_2^2(0)$ is undefined. We formally set undefined values to ∞.

With *the cost functions* of G we refer to the collection of (σ, τ)-cost functions of G for all possible combinations of σ and τ. Let G be the composition of two or more (for a rigid composition) graphs $G_1, \dots, G_\ell$. Computing the cost functions of G assuming that the cost functions of $G_1, \dots, G_\ell$ are known is called *computing cost functions of a composition.* The following theorem states that the ability to compute cost functions of compositions suffices to solve OptimalFlexDraw. The terms T_S, T_P and $T_R(\ell)$ denote the time for computing the cost functions of a series, a parallel, and a rigid composition with skeleton of size ℓ, respectively.

Theorem 2. *Let G be an st-graph containing the edge st. An optimal (σ, τ)-orthogonal representation of G with st on the outer face can be computed in $O(nT_S + nT_P + T_R(n))$ time.*

Applying Theorem 2 for each pair of adjacent nodes as poles in a given instance of OptimalFlexDraw yields the following corollary.

Corollary 1. OptimalFlexDraw *can be solved in $O(n \cdot (nT_S + nT_P + T_R(n)))$ time for biconnected graphs.*

In the following, we extend this result to the case where G may contain cutvertices. The extension is straightforward, however, there is one pitfall. Given two blocks B_1 and B_2 sharing a cutvertex v such that v has degree 2 in B_1 and B_2, we have to ensure for both blocks that v does not form an angle of 180°. Thus, for a given graph G, we get for each block a list of vertices and we restrict the set of all orthogonal representations of G to those where these vertices form 90° angles. We call these orthogonal representations *restricted orthogonal representations.* Moreover, we call the resulting cost functions *restricted cost functions.* We use the terms T_S^r, T_P^r and $T_R^r(\ell)$ to denote the time necessary to compute the *restricted* cost functions of a series, a parallel, and a rigid composition, respectively. We get the following theorem.

Theorem 3. OptimalFlexDraw *is $O(n \cdot (nT_S^r + nT_P^r + T_R^r(n)))$-time solvable.*

Note that Theorem 3 provides a framework for uniform treatment of bend minimization over all planar embeddings in orthogonal drawings. In particular, the polynomial-time algorithm for FlexDraw with positive flexibility [2] can be expressed in this way. There, all resulting cost functions of st-graphs are 0 on a non-empty interval containing 0 (with one minor exception) and ∞, otherwise. Thus, the cost functions of the compositions can be computed using Tamassia's flow network. The results on OptimalFlexDraw [3] can be expressed similarly. When restricting the number of bends of each st-graph occurring in the composition to 3, all resulting cost functions are convex (with one minor exception). Thus, Tamassia's flow network can again be used to compute the cost functions of the compositions. The overall optimality follows from the fact that there exists an optimal solution that can be composed in such a way. In the following sections we see two further applications of this framework.

5 Series-Parallel Graphs

In this section we show that the cost functions of a series composition (Lemma 3) and a parallel composition (Lemma 4) can be computed efficiently. Using our framework, this leads to a polynomial-time algorithm for OptimalFlexDraw for series-parallel graphs with monotone cost functions (Theorem 4). We note that this is only a slight extension to the results by Di Battista et al. [5]. However, it shows the easy applicability of the above framework before diving into the more complicated FPT-algorithm in the following section.

To get the running time claimed in Theorem 4, it is necessary to bound the maximum number ℓ of bends of an st-graph. Using Tamassia's flow network it can be seen that $\ell \in O(n)$.

Lemma 3. *If the (restricted) cost functions of two st-graphs are ∞ for bend numbers larger than ℓ, the (restricted) cost functions of their series composition can be computed in $O(\ell^2)$ time.*

Proof. We first consider the case of non-restricted cost functions. Let G_1 and G_2 be the two st-graphs with poles s_1, t_1 and s_2, t_2, respectively, and let G be their series composition with poles $s = s_1$ and $t = t_2$. For each of the constantly many valid combinations of σ and τ, we compute the (σ, τ)-cost function separately. Assume for the following, that σ and τ are fixed. Since G_1 and G_2 both have at most ℓ bends, G can only have $O(\ell)$ possible values for the number of bends β. We fix the value β and show how to compute $\text{cost}_\tau^\sigma(\beta)$ in $O(\ell)$ time.

Let $\mathcal{R}$ be a (σ, τ)-orthogonal representation with β bends and let $\mathcal{R}_1$ and $\mathcal{R}_2$ be the (σ_1, τ_1)- and (σ_2, τ_2)-orthogonal representations induced for G_1 and G_2, respectively. Obviously, $\sigma_1 = \sigma$ and $\tau_2 = \tau$ holds. However, there are the following other parameters that may vary (although they may restrict each other). The parameters τ_1 and σ_2; the number of bends β_1 and β_2 of $\mathcal{R}_1$ and $\mathcal{R}_2$, respectively; the possibility that for $i \in \{1, 2\}$ the number of bends of $\mathcal{R}_i$ are determined by $\pi(s_i, t_i)$ or by $\pi(t_i, s_i)$, i.e., $\beta_i = -\text{rot}(\pi(s_i, t_i))$ or $\beta_i = -\text{rot}(\pi(t_i, s_i))$; and finally, the rotations at the vertex v in the outer face, where v is the vertex of G belonging to both, G_1 and G_2.

Assume we fixed the parameters τ_1 and σ_2, the choice by which paths β_1 and β_2 are determined, the rotations at the vertex v, and the number of bends β_1 of $\mathcal{R}_1$. Then there is no choice left for the number of bends β_2 of $\mathcal{R}_2$, as choosing a different value for β_2 also changes the number of bends β of G, which was fixed. As each of the parameters can have only a constant number of values except for β_1, which can have $O(\ell)$ different values, there are only $O(\ell)$ possible choices in total. For each of these choices, we get a (σ, τ)-orthogonal representation of G with β bends and cost $\text{cost}_{\tau_1}^{\sigma_1}(\beta_1) + \text{cost}_{\tau_2}^{\sigma_2}(\beta_2)$. By taking the minimum cost over all these choices we get the desired value $\text{cost}_{\tau}^{\sigma}(\beta)$ in $O(\ell)$ time.

For restricted cost functions, we may have to restrict the angle at v. Obviously, this constraint can be easily added to the described algorithm. □

Lemma 4. *If the (restricted) cost functions of two st-graphs are ∞ for bend numbers larger than ℓ, the (restricted) cost functions of their parallel composition can be computed in $O(\ell)$ time.*

Theorem 4. *For series-parallel graphs with monotone cost functions* OptimalFlexDraw *can be solved in $O(n^4)$ time.*

6 An FPT-Algorithm for General Graphs

Let G be an instance of FlexDraw. We call an edge in G *critical* if it is inflexible and at least one of its endpoints has degree 4. We call G k-critical, if it contains exactly k critical edges. An inflexible edge that is not critical is *semi-critical.* The poles s and t of an st-graph G are considered to have additional neighbors (which comes from the fact that we usually consider st-graphs to be subgraphs of larger graphs). Inflexible edges incident to the pole s (or t) are already *critical* if $\deg(s) \geq 2$ (or $\deg(t) \geq 2$). In the following, we study cost functions of k-critical st-graphs and give an FPT-algorithm for k-critical instances.

6.1 The Cost Functions of *k*-Critical Instances

Let G be an st-graph and let $\mathcal{R}$ be a valid orthogonal representation of G. We define an operation that transforms $\mathcal{R}$ into another valid orthogonal representation of G. Let $G^\star$ be the *double directed* dual graph of G, i.e., each edge e of G with incident faces g and f corresponds to the two dual edges (g, f) and (f, g). We call a dual edge $e^\star = (g, f)$ of e *valid* if one of the following conditions holds.

(I) $\text{rot}(e_f) < \text{flex}(e)$ (which is equivalent to $-\text{rot}(e_g) < \text{flex}(e)$).

(II) $\text{rot}(v_f) < 1$ where v is an endpoint of e but not a pole.

A simple directed cycle $C^\star$ in $G^\star$ consisting of valid edges is called *valid cycle.* Then *bending along* $C^\star$ changes the orthogonal representation $\mathcal{R}$ as follows; see Fig. 3a. Let $e^\star = (g, f)$ be an edge in $C^\star$ with primal edge e. If $e^\star$ is valid due to Condition (I), we reduce $\text{rot}(e_g)$ by 1 and increase $\text{rot}(e_f)$ by 1. Otherwise, if Condition (II) holds, we reduce $\text{rot}(v_g)$ by 1 and increase $\text{rot}(v_f)$ by 1, where v is the vertex incident to e with $\text{rot}(v_f) < 1$.

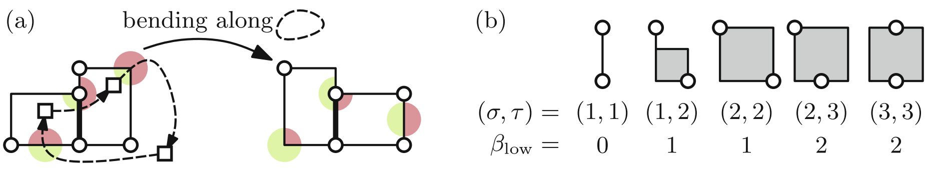
(a)
bending along
(b)
$(\sigma,\tau) =$ (1, 1) (1, 2) (2, 2) (2, 3) (3, 3)
$\beta_{\mathrm{low}} =$ 0 1 1 2 2

Fig. 3. (a) An orthogonal representation (the bold edge is inflexible, other edges have flexibility 1) and a valid cycle $C^\star$ (dashed). Bending along $C^\star$ increases the green and decreases the red angles. (b) Illustration of Fact 1 for some values of σ and τ.

Lemma 5. *Let G be an st-graph with a valid (σ, τ)-orthogonal representation $\mathcal{R}$. Bending along a valid cycle $C^\star$ yields a valid (σ, τ)-orthogonal representation.*

As mentioned in Section 4, depending on σ and τ, there is a lower bound β_{low} on the number of bends of (σ, τ)-orthogonal representations; see Fig. 3b.

Fact 1. *A (σ, τ)-orthogonal representation has at least $\beta_{\mathrm{low}} = \left\lceil \frac{\sigma+\tau}{2} \right\rceil - 1$ bends.*

For a valid orthogonal representation with a large number of bends, the following lemma states that we can reduce its bends by bending along a valid cycle. This can later be used to show that the cost function of an *st*-graph is 0 on a significantly large interval. Or in other words, arbitrary alterations of cost 0 and cost ∞ that are hard to handle only occur on a small interval (depending on k). The lemma and its proof are a generalization of Lemma 1 from [2] that incorporates inflexible edges. For $\sigma = \tau = 3$ a slightly weaker result holds.

Lemma 6. *Let G be a k-critical st-graph and let $\mathcal{R}$ be a valid (σ, τ)-orthogonal representation with $\sigma + \tau \leq 5$. If $-\operatorname{rot}(\pi(t, s)) \geq \beta_{\mathrm{low}} + k + 1$ holds, then there exists a valid cycle $C^\star$ such that bending $\mathcal{R}$ along $C^\star$ reduces $-\operatorname{rot}(\pi(t, s))$ by 1.*

Lemma 7. *Let G be a k-critical st-graph and let $\mathcal{R}$ be a valid $(3, 3)$-orthogonal representation. If $-\operatorname{rot}(\pi(t, s)) \geq \beta_{\mathrm{low}} + k + 2$ holds, then there exists a valid cycle $C^\star$ such that bending $\mathcal{R}$ along $C^\star$ reduces $-\operatorname{rot}(\pi(t, s))$ by 1.*

The previous lemmas basically show that the existence of a valid orthogonal representation with a lot of bends implies the existence of valid orthogonal representations for a "large" interval of bend numbers. This is made more precise in the following.

Let $\mathcal{B}_\tau^\sigma$ be the set containing an integer β if and only if G admits a valid (σ, τ)-orthogonal representation with β bends. Assume G admits a valid (σ, τ)-orthogonal representation, i.e., $\mathcal{B}_\tau^\sigma$ is not empty. We define the *maximum bend value* $\beta_{\max}$ to be the maximum in $\mathcal{B}_\tau^\sigma$. Moreover, let $\beta \in \mathcal{B}_\tau^\sigma$ be the smallest value, such that every integer between β and $\beta_{\max}$ is contained in $\mathcal{B}_\tau^\sigma$. Then we call the interval $[\beta_{\mathrm{low}}, \beta - 1]$ the (σ, τ)-*gap* of G. The value $\beta - \beta_{\mathrm{low}}$ is also called the (σ, τ)-*gap* of G; see Fig. 4.

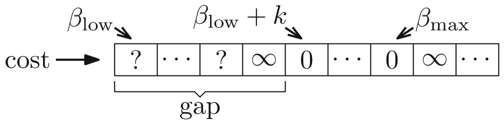

Fig. 4. A cost function with gap k

Lemma 8. *The (σ, τ)-gap of a k-critical st-graph G is at most k if $\sigma + \tau \leq 5$. The $(3, 3)$ gap of G is at most $k + 1$.*

The following lemma basically expresses the gap of an *st*-graph in terms of the rotation along $\pi(s, t)$ instead of the number of bends.

Lemma 9. *Let G be an st-graph with (σ, τ)-gap k. The set $\{\rho \mid G$ admits a valid (σ, τ)-orthogonal representation with $\operatorname{rot}(\pi(s, t)) = \rho\}$ is the union of at most $k + 1$ intervals.*

6.2 Computing the Cost Functions of Compositions

Let G be a graph with fixed planar embedding. We describe a flow network, similar to the one by Tamassia [9] that can be used to compute orthogonal representations of graphs with thick edges. In general, we consider a flow network to be a directed graph with a lower and an upper bound assigned to every edge and a demand assigned to every vertex. The bounds and demands can be negative. An assignment of flow-values to the edges is a feasible flow if it satisfies the following properties. The flow-value of each edge is at least its lower and at most its upper bound. For every vertex the flow on incoming edges minus the flow on outgoing edges must equal its demand.

We define the flow network N as follows. The network N contains a node for each vertex of G, the *vertex nodes*, each face of G, the *face nodes*, and each edge of G, the *edge nodes*. Moreover, N contains arcs from each vertex to all incident faces, the *vertex-face arcs*, and similarly from each edge to both incident faces, the *edge-face arcs*. We interpret an orthogonal representation $\mathcal{R}$ of G as a flow in N. A rotation $\operatorname{rot}(e_f)$ of an edge e in the face f corresponds to the same amount of flow on the edge-face arc from e to f. Similarly, for a vertex v incident to f the rotation $\operatorname{rot}(v_f)$ corresponds to the flow from v to f.

Obviously, the properties (1)–(4) of an orthogonal representation are satisfied if and only if the following conditions hold for the flow (note that we allow G to have thick edges).

(1) The total amount of flow on arcs incident to a face node is 4 (-4 for the outer face).
(2) The flow on the two arcs incident to an edge node stemming from a (σ, τ)-edge sums up to $2 - (\sigma + \tau)$.
(3) The total amount of flow on arcs incident to a vertex node, corresponding to the vertex v with incident edges $e_1, \dots, e_\ell$ occupying $\sigma_1, \dots, \sigma_\ell$ incidences of v is $\sum(\sigma_i + 1) - 4$.
(4) The flow on vertex-face arcs lies in the range $[-2, 1]$.

Properties (1)–(3) are equivalent to the flow conservation requirement when setting appropriate demands. Property (4) is equivalent to the capacity constraints in a flow network when setting the lower and upper bounds of vertex-face arcs to -2 and 1, respectively. In the following, we use this flow network to compute the cost function of a rigid composition of graphs. The term $T_{\text{flow}}(\ell)$ denotes the time necessary to compute a maximal flow in a planar flow network of size ℓ.

Lemma 10. *The (restricted) cost functions of a rigid composition of ℓ graphs can be computed in $O(2^k \cdot T_{\text{flow}}(\ell))$ time if the resulting graph is k-critical.*

Proof (sketch). Let H be the skeleton of the rigid composition of the graphs $G_1, \dots, G_\ell$ and let G be the resulting graph with poles s and t. In the following, we show that the (restricted) cost functions can be determined by computing flows in $O(2^k)$ flow networks. We first show that, similar to the proof of Lemma 3, the following parameters lead to only constantly many combinations: the number of occupied incidences of the composed graphs and the graph itself; the decision whether $\pi(s,t)$ or $\pi(t,s)$ determines the number of bends of G. The remaining degrees of freedom are the number of bends β of G and for each graph G_i an interval for its rotation along $\pi(s_i,t_i)$, where s_i and t_i are the poles of G_i. It is not hard to see that all these constraints can be expressed by capacity constraints in the flow network. It remains to count the number of combinations.

Let k_i be the number of critical edges in G_i. Due to Lemma 9, we get $k_i + 1$ intervals for each of these graphs G_i leading to $\prod(k_i + 1)$ combinations. As critical edges in G_i remain critical in G (and the G_i are pairwise edge-disjoint), we have $\sum k_i \leq k$. One can show that with this restriction $\prod(k_i + 1) \in O(2^k)$.

Constructing all these flow networks for every possible number of bends β would lead to $O(\beta_{\max} \cdot 2^k)$ flow networks. However, once the rotation intervals for the G_i are fixed, one can compute a maximum and minimum rotation for G with these intervals. It follows from basic flow theory that all intermediate rotation values are also possible. Thus with two flow computations, we obtain all possible rotation values for the chosen intervals of the G_i. This leads to $O(2^k)$ intervals whose union describes the cost function of G. From that, an explicit representation of the cost function of G can be computed in $O(2^k)$ time, which yields a total running time of $O(2^k \cdot T_{\text{flow}}(\ell))$. □

Lemma 11. *The (restricted) cost functions of a series and a parallel composition can be computed in $O(k^2 + 1)$ time if the resulting graph is k-critical.*

Theorem 5. FlexDraw *for k-critical graphs is $O(2^k \cdot n \cdot T_{\text{flow}}(n))$-time solvable.*

Proof. By Theorem 3, we get an algorithm with the running time $O(n \cdot (n \cdot T_S + n \cdot T_P + T_R(n)))$, where $T_S, T_P \in O(k^2 + 1)$ (Lemma 11) and $T_R(\ell) = 2^k \cdot T_{\text{flow}}(\ell)$ (Lemma 10) holds. This obviously yields the running time $O((k^2+1) \cdot n^2 + 2^k \cdot n \cdot T_{\text{flow}}(n)) = O(2^k \cdot n \cdot T_{\text{flow}}(n))$. □

7 Conclusion

We want to conclude with the open question whether there exists an FPT-algorithm for OptimalFlexDraw for the case where all cost functions are convex and where the first bend causes cost only for k edges. One might think that this works similar as for FlexDraw by showing that the cost functions of st-graphs are only non-convex if they contain inflexible edges. Then, when encountering a rigid composition, one could separate these non-convex cost functions into convex parts and consider all combinations of these convex parts. Unfortunately, the cost functions of st-graphs may already be non-convex, even though they do not contain inflexible edges. The reason why OptimalFlexDraw can still be solved efficiently if there are no inflexible edges [3] is that, in this case, the cost functions need to be considered only up to three bends (and for this restricted intervals, the cost functions are convex). However, a single subgraph with inflexible edges in a rigid composition may force arbitrary other subgraphs in this composition to have more than three bends, potentially resulting in linearly many non-convex cost functions that have to be considered. Thus, although the algorithms for FlexDraw and OptimalFlexDraw are very similar, the latter does not seem to allow even a small number of inflexible edges.

Acknowledgments. We thank Marcus Krug for discussions on FlexDraw.

References

1. Biedl, T., Kant, G.: A better heuristic for orthogonal graph drawings. Comput. Geom.: Theory Appl. **9**(3), 159–180 (1998)
2. Bläsius, T., Krug, M., Rutter, I., Wagner, D.: Orthogonal graph drawing with flexibility constraints. Algorithmica 68(4) (2014)
3. Bläsius, T., Lehmann, S., Rutter, I.: Orthogonal graph drawing with inflexible edges. CoRR abs/1404.2943, 1–23 (2014). http://arxiv.org/abs/1404.2943
4. Bläsius, T., Rutter, I., Wagner, D.: Optimal orthogonal graph drawing with convex bend costs. In: Fomin, F.V., Freivalds, R., Kwiatkowska, M., Peleg, D. (eds.) ICALP 2013, Part I. LNCS, vol. 7965, pp. 184–195. Springer, Heidelberg (2013)
5. Cornelsen, S., Karrenbauer, A.: Accelerated bend minimization. J. Graph Alg. Appl. **16**(3), 635–650 (2012)
6. Di Battista, G., Liotta, G., Vargiu, F.: Spirality and optimal orthogonal drawings. SIAM J. Comput. **27**(6), 1764–1811 (1998)
7. Di Battista, G., Tamassia, R.: On-line maintenance of triconnected components with SPQR-trees. Algorithmica **15**(4), 302–318 (1996)
9. Garg, A., Tamassia, R.: On the computational complexity of upward and rectilinear planarity testing. SIAM J. Comput. **31**(2), 601–625 (2001)
9. Tamassia, R.: On embedding a graph in the grid with the minimum number of bends. SIAM J. Comput. **16**(3), 421–444 (1987)