Computational Geometry 55 (2016) 26–40

Contents lists available at ScienceDirect

Computational Geometry: Theory and Applications

www.elsevier.com/locate/comgeo

Orthogonal graph drawing with inflexible edges ☆, ☆☆

Thomas Bläsius [a,b,*], Sebastian Lehmann [a], Ignaz Rutter [a,*]

[a] Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany
[b] Hasso Plattner Institute, Potsdam, Germany

ARTICLE INFO

Article history:
Received 19 June 2015
Received in revised form 23 December 2015
Accepted 26 February 2016
Available online 3 March 2016

Keywords:
Orthgonal graph drawing
Bend minimization
Planar embedding
Parameterized algorithm
Computational complexity

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 [1]. On the other hand, FlexDraw can be solved efficiently if $\text{flex}(e) \geq 1$ [2] and is trivial if $\text{flex}(e) \geq 2$ [3] 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), where n denotes the number of vertices in the graph. 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 [1] 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 planar graphs with maximum degree 3 and series-parallel graphs, Di Battista et al. [4]

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

☆☆ A preliminary version of this paper has appeared as T. Bläsius, S. Lehmann, I. Rutter, Orthogonal graph drawing with inflexible edges, in: Proceedings of the 9th International Conference on Algorithms and Complexity, in: Lecture Notes in Computer Science, vol. 9070, Springer, 2015, pp. 153–166.

* Corresponding authors.
E-mail addresses: thomas.blaesius@hpi.de (T. Bläsius), sebastian@leemes.de (S. Lehmann), rutter@kit.edu (I. Rutter).

http://dx.doi.org/10.1016/j.comgeo.2016.03.001
0925-7721/

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 [3] 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. [2,5] 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 [5].

When restricting the allowed drawings to those with a specific planar embedding, the problem OptimalFlexDraw becomes significantly easier. Tamassia [6] 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 [7] recently showed, that this kind of flow network can be solved in $O(n^{3/2})$ time.

Contribution & outline In this work we consider OptimalFlexDraw for instances that may contain inflexible edges, closing the gap between the general NP-hardness result [1] and the polynomial-time algorithms in the absence of inflexible edges [2,5]. 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 non-decreasing cost functions. This extends the algorithm by Di Battista et al. [4] 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. [4]. We conclude with open questions in Section 7.

2. Preliminaries

2.1. Connectivity & 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 such that $G + st$ is biconnected and planar. The vertices s and t are called the *poles* of G. 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. Moreover, 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. Then 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 [8,9].

2.2. SPQR-tree

The SPQR-tree $\mathcal{T}$ of a biconnected st-graph G containing the edge st is a rooted tree encoding series, parallel and rigid compositions of st-graphs that result in the graph G [8,9]. The leaves of $\mathcal{T}$ are *Q-nodes* representing the edges of G and thus the st-graphs we start with. The root of $\mathcal{T}$ is also a Q-node, representing the special edge st. Each inner node is either an *S-node*, representing one or more series compositions of its children, a *P-node*, representing one or more parallel compositions of its children, or an *R-node*, representing a rigid composition of its children.

Recall that the rigid composition is performed with respect to a skeleton. For an R-node μ, let H be the skeleton of the corresponding rigid composition with poles s_μ and t_μ. We call $H + s_\mu t_\mu$ the *skeleton* of the μ and denote it by $\text{skel}(\mu)$.

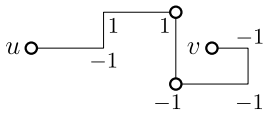

Fig. 1. Illustration of the rotation of a path from u to v. Right bends are labeled 1, left bends are labeled -1. The rotation of the path is -2, which is obtained by summing up all labels.

The special edge $s_\mu t_\mu$ is called *parent edge*, all other edges are virtual edges, each corresponding to one child of μ. We also add skeletons to the other nodes. For an S-node μ, the skeleton $\text{skel}(\mu)$ is a path of virtual edges (one for each child) from s_μ to t_μ together with the parent edge $s_\mu t_\mu$. The skeleton of a P-node μ is a bunch of parallel virtual edges (one for each child) between s_μ and t_μ together with the parent edge $s_\mu t_\mu$. The skeleton of a Q-node contains the edge it represents in G together with a parallel parent edge. The root representing st has no parent edge, thus this additional edge is a virtual edge corresponding to the unique child of the root.

When not allowing pairs of adjacent S-nodes and pairs of adjacent P-nodes in $\mathcal{T}$, then the SPQR-tree is unique for a fixed edge st in G. Moreover, using the endpoints of a different edge as poles of G results in the same SPQR-tree with a different root (the parent edge in each skeleton may also change). For fixed poles s and t, there is a bijection between the planar embeddings of G with st on the outer face and the combinations of embeddings of all skeletons with their parent edges on the outer face. The *pertinent graph* $\text{pert}(\mu)$ of a node μ of $\mathcal{T}$ is recursively defined to be the skeleton $\text{skel}(\mu)$ without the parent edge $s_\mu t_\mu$ after the replacement of every virtual edge with the pertinent graph of the corresponding child. Note that the pertinent graph of the root is G itself. The SPQR-tree can be computed in linear time [10].

2.3. 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 [6], however, we use a slight modification that makes it easier to work with, as bends of edges and bends at vertices are handled in the same way. Let Γ be a *normalized* orthogonal drawing of G, that is 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 [6]. We assume that all orthogonal drawings we consider are normalized.

For the purpose of the following definitions, think of G being biconnected. We assume that G is biconnected. This simplifies the description, as each edge and vertex has at most one incidence to a face. For connected graphs, referring to the incidence of a vertex or an edge and a face may be ambiguous. However, it will be always clear from the context, which incidence is meant.

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 to be $\text{rot}(e_f) = \beta$ and $\text{rot}(e_f) = -\beta$ 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. 2a.

(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 [6].

Orthogonal representations and bends of st-graphs We extend the notion of rotation to paths; conceptually this is very similar to spirality [4]. 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. Note that the rotation counts both bends at vertices and bends on edges; see Fig. 1 for an example.

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. Let $\text{rot}(s)$ and $\text{rot}(t)$ denote the rotations of s and t in the outer face, respectively. We will frequently use the fact that by property (1) it is $\text{rot}(\pi(s,t)) + \text{rot}(t) + \text{rot}(\pi(t,s)) + \text{rot}(s) = -4$. We define the *number of bends* of $\mathcal{R}$ to be $\max\{|\text{rot}(\pi(s,t))|, |\text{rot}(\pi(t,s))|\}$. Note 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. Further, since $\text{rot}(s) \geq -2$ and $\text{rot}(t) \geq -2$, we have $\text{rot}(\pi(s,t)) + \text{rot}(\pi(t,s)) \leq 0$, and therefore $\max\{|\text{rot}(\pi(s,t))|, |\text{rot}(\pi(t,s))|\} \in$

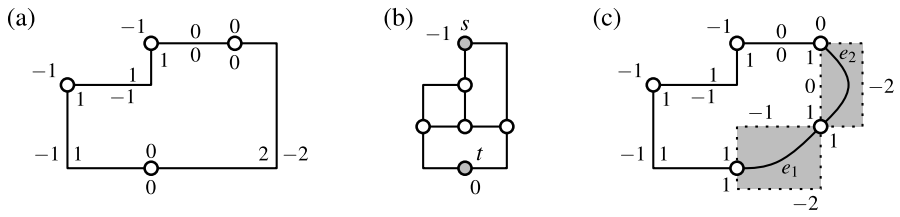

Fig. 2. (a) An orthogonal drawing together with its orthogonal representation given by the rotation values. (b) A $(2,3)$-orthogonal representation (s and t have 2 and 1 free incidences, respectively) with two bends (defined by $-\operatorname{rot}(t,s)$). (c) An orthogonal representation with thick 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.

$\{-\operatorname{rot}(\pi(s,t)), -\operatorname{rot}(\pi(t,s))\}$, i.e., the number of bends of an st-graph is defined by the negative rotation of one of its boundary paths.

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 $\operatorname{rot}(s_f) = \sigma - 3$ 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. 2b.

Note that $\operatorname{rot}(\pi(s,t))$ and $\operatorname{rot}(\pi(t,s))$ 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, that is an edge enters each of its endpoint from exactly one of four possible directions. 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 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. 2c.

(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 $\max\{|\operatorname{rot}(e_{f_\ell})|, |\operatorname{rot}(e_{f_r})|\}$. 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]. This is illustrated in Fig. 2, where replacing the $(2,3)$-edge e_1 in (c), which has two bends, by the $(2,3)$-orthogonal representation of the graph in (b), which also has two bends, yields a valid orthogonal representation.

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, that is they have pairwise distance $\Omega(n^{1-\varepsilon})$. This for example shows NP-hardness for instances with $O(\sqrt{n})$ inflexible edges with pairwise distances of $\Omega(\sqrt{n})$.

We adapt the proof of NP-hardness by Garg and Tamassia [1] 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 [1, Lemma 5.1] and replacing the edges of F by special st-graphs, the so called tendrils and wiggles. Both, tendrils and wiggles, have degree 1 at both poles and a unique planar embedding up to possibly a flip. It follows for each vertex v of G, that the cyclic order of incident edges around v is fixed up to a flip. This implies the following lemma.

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

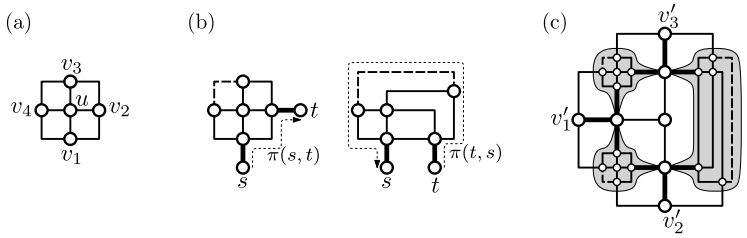

Fig. 3. The bold edges are inflexible; dashed edges have flexibility 2; all other edges have flexibility 1. (a) The wheel W_4. (b) The bend gadget $B_{1,2}$. (c) The gadget W'_3 for replacing degree-3 vertices. The marked subgraphs are bend gadgets.

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. We start with the description of an *st*-graph that has either 1 or 2 bends in every valid orthogonal representation.

The wheel W_4 of size 4 consists of a 4-cycles $v_1, \dots, v_4$ together with a center u connected to each of the vertices $v_1, \dots, v_4$; see Fig. 3a. We add the two vertices s and t together with the inflexible edges sv_1 and tv_2 to W_4. Moreover, we set the flexibility of v_3v_4 to 2 and the flexibilities of all other edges to 1. We call the resulting *st*-graph *bend gadget* and denote it by $B_{1,2}$. We only consider embeddings of $B_{1,2}$ where all vertices except for u lie on the outer face. Fig. 3(b) shows two valid orthogonal representations of $B_{1,2}$, one with 1, the other with 2 bends. Clearly, the number of bends cannot be reduced to 0 (or increased above 2) without violating the flexibility constraints of edges on the path $\pi(s,t)$ (or on the path $\pi(t,s)$). Thus, $B_{1,2}$ has either 1 or 2 bends in every orthogonal representation. Moreover, if its embedding is fixed, then the direction of the bends is also fixed.

We now use the bend gadget as building block for a larger gadget. We start with the wheel W_3 of size 3 consisting of a triangle v_1, v_2, v_3 together with a center u connected to v_1, v_2, and v_3. The flexibilities of the edges incident to the center are set to 1, each edge in the triangle is replaced by a bend gadget $B_{1,2}$. To fix the embedding of the bend gadgets, we add three vertices v'_1, v'_2, and v'_3 connected with inflexible edges to v_1, v_2, and v_3, respectively, and connect them to the free incidences in the bend gadgets, as shown in Fig. 3(c). We denote the resulting graph by W'_3. Clearly, in the cycle of bend gadgets, two of them have one bend and the other has two bends in every valid orthogonal representation of W'_3. Thus, replacing a vertex v with incident edges e_1, e_2, and e_3 by W'_3, attaching the edge e_i to v'_i, yields an equivalent instance of FlexDraw. Note that such a replacement increases the degree of one incidence of e_1, e_2, and e_3 from 3 to 4. Moreover, every inflexible edge contained in W'_3 is incident to two vertices of 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.*

Proof. Let G be an instance of FlexDraw such that the order of edges around each vertex is fixed up to reversal. As FlexDraw restricted to these kinds of instances is NP-hard, due to Lemma 1, it suffices to find an equivalent instance where additionally the endpoints of each inflexible have degree 4. Pairs of edges incident to a vertex of degree 2 can be simply replaced by an edge with higher flexibility. Thus, we can assume that every vertex in G has degree 3 or degree 4. Replacing every degree-3 vertex incident to an inflexible edge by the subgraph W'_3 described above clearly leads to an equivalent instance with the desired properties. □

Similar to the replacement of degree-3 vertices by W'_3, we can replace degree-4 vertices by the wheel W_4, setting the flexibility of every edge of W_4 to 1. It is easy to see, that every valid orthogonal representation of W_4 has the same outer shape, that is a rectangle, with one of the vertices $v_1, \dots, v_4$ on each side; see Fig. 3(a). Thus, replacing a vertex v with incident edges $e_1, \dots, e_4$ (in this order) by W_4, attaching $e_1, \dots, e_4$ to the vertices $v_1, \dots, v_4$ yields an equivalent instance of FlexDraw. This allows us to arbitrarily increase the distance between the edges with flexibility 0, where the distance between two edges e and f is the length of a shortest-path connecting and endpoint of e to an endpoint of f.

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

Proof. As FlexDraw is clearly in NP, it remains to show NP-hardness. Let G be the instance of FlexDraw such that the endpoints of each inflexible edge have degree 4 and such that the order of edges around each vertex is fixed up to reversal. FlexDraw restricted to these kinds of instances is NP-hard due to Lemma 2. We show how to build an equivalent instance with $O(n^\varepsilon)$ inflexible edges with pairwise distance $\Omega(n^{1-\varepsilon})$ for any $\varepsilon > 0$.

Let e be an inflexible edge in G with incident vertices u and v, which both have degree 4. Replacing each of the vertices u and v by the wheel W_4 yields an equivalent instance of FlexDraw and the distance of e to every other inflexible edge

is increased by a constant. Note that this does not increase the number of inflexible edges. Let n_G be the number of vertices in G. Applying this replacement $n_G^{1/\varepsilon-1}$ times to the vertices incident to each inflexible edge yields an equivalent instance G'. In G' every pair of inflexible edges has distance $\Omega(n_G^{1/\varepsilon-1})$. Moreover, G' has size $O(n_G^{1/\varepsilon})$, as we have n_G inflexible edges. Substituting $n_G^{1/\varepsilon}$ by n shows that we get an instance of size n with $O(n^\varepsilon)$ inflexible edges with pairwise distance $\Omega(n^{1-\varepsilon})$. □

Note that the instances described above may contain edges with flexibility larger than 1. We can get rid of that as follows. An edge $e = st$ with flexibility $\text{flex}(e) > 0$ can have the same numbers of bends as the st-graph consisting of the wheel W_4 (Fig. 3(a)) with the additional edges sv_1 and tv_3, where $\text{flex}(sv_1) = 1$ and $\text{flex}(tv_3) = \text{flex}(e) - 1$. Thus, we can successively replace edges with flexibility above 1 by these kinds of subgraphs, leading to an equivalent instance where all edges have flexibility 0 or 1.

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, which is straightforward. 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 an st-graph G, we refer to the collection of (σ,τ)-cost functions of G for all possible combinations of σ and τ.

The main idea for our algorithm that computes an optimal orthogonal drawing of a graph H with a fixed edges $e = st$ on the outer face is as follows. We take the SPQR-tree $\mathcal{T}$ of H rooted at st, and we compute the cost functions of its pertinent graphs, in a bottom-up fashion. Note that each pertinent graph is an st-graph, and they are combined by a series, a parallel, or a rigid composition.

Let the st-graph G be the composition of two or more (for a rigid composition) st-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 necessary to compute the cost functions of a series, a parallel, and a rigid composition with skeleton of size ℓ, respectively. Recall that series and parallel compositions always compose only two graphs, while a rigid composition composes ℓ graphs, where ℓ is the size of the corresponding skeleton (therefore, T_R depends on ℓ, while T_S and T_P do not). Note that S-nodes and P-nodes in the SPQR-tree may give rise to a sequence of series and parallel compositions, respectively.

Theorem 2. *Let G be an st-graph with n vertices that contains 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.*

Proof. Let $\mathcal{T}$ be the SPQR-tree of G. To compute an optimal orthogonal representation of G with st on the outer face, we root $\mathcal{T}$ at the Q-node corresponding to st and traverse it bottom up. When processing a node μ, we compute the cost functions of $\text{pert}(\mu)$, which finally (in the root) yields the cost functions of the st-graph G and thus optimal (σ,τ)-orthogonal representations (for all possible values of σ and τ) with st on the outer face.

If μ is a **Q-node** but not the root, then $\text{pert}(\mu)$ is an edge and the cost function of this edge is given with the input.

If μ is an **S-node**, its pertinent graph can be obtained by applying multiple series compositions. Since the skeleton of an S-node leaves no embedding choice, we can compute the cost function of $\text{pert}(\mu)$ by successively computing the cost functions of the compositions, which takes $O(|\text{skel}(\mu)| \cdot T_S)$ time.

If μ is a **P-node**, then $\text{pert}(\mu)$ can be obtained by applying multiple parallel compositions. In contrast to S-nodes the skeleton of a P-node leaves an embedding choice, namely changing the order of the parallel edges. As composing the pertinent graphs of the children of μ in a specific order restricts the embedding of $\text{skel}(\mu)$, we cannot apply the compositions in an arbitrary order if $\text{skel}(\mu)$ contains more than two parallel edges (not counting the parent edge). However, since $\text{skel}(\mu)$ contains at most three parallel edges (due to the restriction to degree 4), we can try all composition orders and take the minimum over the resulting cost functions. As there are only constantly many orders and for each order a constant number of compositions is performed, computing the cost function of $\text{pert}(\mu)$ takes $O(T_P)$ time.

If μ is an **R-node**, the pertinent graph of μ is the rigid composition of the pertinent graphs of its children with respect to the skeleton $\text{skel}(\mu)$. Thus, the cost functions of $\text{pert}(\mu)$ can be computed in $O(T_R(|\text{skel}(\mu)|))$ time.

If μ is the **root**, that is the Q-node corresponding to st, then $\text{pert}(\mu) = G$ is a parallel composition of the pertinent graph of the child of μ and the edge st and thus its cost function can be computed in $O(T_P)$ time.

As the total size of S-node skeletons, the number of P-nodes and the total size of R-node skeletons is linear in the size of G, the running time is in $O(n \cdot T_S + n \cdot T_P + T_R(n))$. □

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 obtain 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 obtain the following extension of the previous results.

Theorem 3. OptimalFlexDraw *can be solved in* $O(n \cdot (nT_S^r + nT_P^r + T_R^r(n)))$ *time.*

Proof. Let G be an instance of OptimalFlexDraw. We use the BC-tree (Block–Cutvertex Tree) of G to represent all possible ways of combining embeddings of the blocks of G to an embedding of G. The *BC-tree* $\mathcal{T}$ of G contains a B-node for each block of G, a C-node for each cutvertex of G and an edge between a C-node and a B-node if and only if the corresponding cutvertex is contained in the corresponding block, respectively.

Rooting $\mathcal{T}$ at some B-node restricts the embeddings of the blocks as follows. Let μ be a B-node (but not the root) corresponding to a block B and let v be the cutvertex corresponding to the parent of μ. Then the embedding of B is required to have v on its outer face. It is easy to see that every embedding of G is such a restricted embedding with respect to some root of $\mathcal{T}$. Thus, it suffices to consider each B-node of $\mathcal{T}$ as root and restrict the embeddings as described above.

Before we deal with the BC-tree $\mathcal{T}$, we preprocess each block B of G. Let v be a cutvertex of B. For an edge e incident to v, we can use Theorem 2 to compute an optimal orthogonal representation of B with e on the outer face in $O(n \cdot T_S + n \cdot T_P + T_R(n))$ time. Since every orthogonal representation with v on the outer face has one of its incident edges on the outer face, we can simply force each of these edges to the outer face once, to get an optimal orthogonal representation of B with v on the outer face. Clearly, using the computation of restricted cost functions yields an optimal restricted orthogonal representation. Doing this for each block of G and for each cutvertex in this block leads to a total running time of $O(n \cdot (n \cdot T_S + n \cdot T_P + T_R(n)))$. Moreover, we can compute an optimal restricted orthogonal representation of each block (without forcing a vertex to the outer face) with the same running time (Corollary 1).

To compute an optimal orthogonal representation of G we choose every B-node of the BC-tree $\mathcal{T}$ as root and consider for the block corresponding to the root the optimal orthogonal representation (without forcing vertices to the outer face). For all other blocks we consider the optimal orthogonal representation with the cutvertex corresponding to its parent on the outer face. Note that these orthogonal representations can be easily combined to an orthogonal representation of the whole graph, as we enforce angles of 90° at vertices of degree 2, if they have degree 2 in another block. The minimum over all roots leads to an optimal orthogonal representation. As computing this minimum takes $O(n^2)$ time, it is dominated by the running time necessary to compute the orthogonal representation of the blocks. □

Note that Theorem 3 provides a framework for uniform treatment of bend minimization over all planar embeddings in orthogonal drawings. To obtain an algorithm for a specific class of instances, it suffices to give efficient algorithms for computing series, parallel, and rigid compositions of these instances.

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 [5] 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, resulting in efficient algorithms.

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 non-decreasing cost functions (Theorem 4). We note that this is only a slight extension to the results by Di Battista et al. [4]. However, it shows the easy applicability of the above framework before diving into the more complicated FPT-algorithm in the following section.

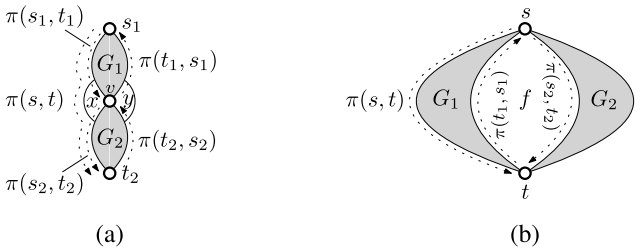

Fig. 4. Illustration of the proofs of Lemma 3 (a) and Lemma 4 (b).

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. Without loss of generality, we assume that $\beta = -\text{rot}(\pi(s,t))$, otherwise we mirror $\mathcal{R}$. 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 $\pi(t_i,s_i)$, that is $\beta_i = -\text{rot}(\pi(s_i,t_i))$ 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; see Fig. 4a.

Assume we fixed the parameters τ_1 and σ_2, the choice by which paths β_1 and β_2 are determined, the rotations x and y at the vertex v lying on $\pi(s,t)$ and $\pi(t,s)$, respectively, and the number of bends β_1 of $\mathcal{R}_1$. Note that not all combinations let us actually combine the orthogonal representations. More precisely, they can be combined if and only if $\tau_1 + \sigma_2 + (2 - x - y) = 4$, which can be seen as follows. The rotations x and y indicate that v has $(1-x)+(1-y)$ free incidences in the outer face. Clearly, the number of incidences occupied by $\mathcal{R}_1$ and $\mathcal{R}_2$ together with the free incidences in the outer face has to be 4.

Once the above parameters are fixed, 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 assumed to be fixed. More precisely, we compute β_2 as follows. We have that the total rotation around the outer face of each $\mathcal{R}_i$ is -4, and therefore, for $i = 1, 2$, we have

$$\text{rot}(\pi(s_i,t_i)) + \text{rot}(\pi(t_i,s_i)) + (\sigma_i - 3) + (\tau_i - 3) = -4. \tag{1}$$

If $\text{rot}(\pi(s_1,t_1)) \neq -\beta_1$, then we have $\text{rot}(\pi(t_1,s_1)) = -\beta_1$, and Equation (1) allows us to compute $\text{rot}(\pi(s_1,t_1))$. We then compute $\text{rot}(\pi(s_2,t_2))$ using the fact that $\text{rot}(\pi(s,t)) = \text{rot}(\pi(s_1,t_1)) + x + \text{rot}(\pi(s_2,t_2)) = -\beta$. This either immediately yields β_2 (if β_2 is defined by $-\text{rot}(\pi(s_2,t_2))$), or we use again Equation (1) to compute $\beta_2 = -\text{rot}(\pi(t_2,s_2))$.

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.

If we consider restricted cost functions, it may happen that the vertex v has degree 2. Then we need to enforce an angle of 90° there. 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.*

Proof. Similar to the proof of Lemma 3, we consider several parameters and show that we have to check $O(\ell)$ combinations to obtain the cost function of the composition G. As before, 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 representation induced for G_1 and G_2, respectively. We assume that $\beta = -\text{rot}(\pi(s,t))$ and that G_1 lies to the left of G_2 (when going from s to t); the other cases are symmetric.

We fix the values β, σ, and τ and show how to compute $\text{cost}_\tau^\sigma(\beta)$ in constant time. We will see later that we have to consider only $O(\ell)$ different values for β (and clearly only a constant number of values for σ and τ). To obtain $\text{cost}_\tau^\sigma(\beta)$, assume that (in addition to β, σ, and τ), we fixed values for the parameters σ_1, τ_1, σ_2, and τ_2 (leading to constantly many combinations). We show that this already determines all other relevant parameters of $\mathcal{R}_1$ and $\mathcal{R}_2$, namely $\text{rot}(\pi(s_1,t_1))$, $\text{rot}(\pi(t_1,s_1))$, $\text{rot}(\pi(s_2,t_2))$, and $\text{rot}(\pi(t_2,s_2))$; see Fig. 4b. Thus, the number of bends β_1 and β_2 of G_1 and G_2 are also

determined and we can look up the cost using the cost function for G_1 and G_2. The minimum cost we obtain this way is then the desired value $\text{cost}_\tau^\sigma(\beta)$.

We start with $\text{rot}(\pi(s_1,t_1))$. As G_1 lies to the left of G_2, we have $\pi(s_1,t_1) = \pi(s,t)$. Thus, $\text{rot}(\pi(s_1,t_1)) = \text{rot}(\pi(s,t)) = -\beta$. To obtain the rotation of the other path of G_1, namely $\text{rot}(\pi(t_1,s_1))$, we can use Equation (1) from the previous lemma.

To obtain the rotations of the paths of G_2, consider the inner face f between G_1 and G_2. It is bounded by the two paths $\pi(t_1,s_1)$ and $\pi(s_2,t_2)$. As the total rotation in f is 4, we have $\text{rot}(\pi(t_1,s_1)) + \text{rot}(s_f) + \text{rot}(\pi(s_2,t_2)) + \text{rot}(t_f) = 4$. Thus, if we know the rotations at s and t in f, this gives us the rotation of $\pi(s_2,t_2)$. To obtain $\text{rot}(s_f)$, first note that we can dismiss the current choice of parameters if $\sigma_1 + \sigma_2 \leq \sigma$ does not hold (G cannot occupy fewer incidences than G_1 and G_2 together). Let $\Delta_s = \sigma - \sigma_1 - \sigma_2$. Clearly, the vertex s must have Δ_s free incidences in the face f, i.e., the rotation at s in f is $\Delta_s - 1$. The same arguments let us determine $\text{rot}(t_f)$ and thus, with the above arguments, $\text{rot}(\pi(s_2,t_2))$. Again using Equation (1) from the previous lemma lets us determine $\text{rot}(\pi(t_2,s_2))$.

Finally, note that $\beta \leq \beta_1$, as $\beta = -\text{rot}(\pi(s,t)) = -\text{rot}(\pi(s_1,t_1)) \leq \beta_1$. Thus, the cost function $\text{cost}_\tau^\sigma(\beta)$ is ∞ if $\beta > \ell$, which shows that it suffices to consider only $O(\ell)$ values for β. □

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

Proof. To solve OptimalFlexDraw, we use Theorem 3. As the graphs we consider here are series-parallel, it suffices to give algorithms that compute the cost functions of series and parallel compositions. Applying Lemma 3 and Lemma 4 gives us running times $T_S \in O(\ell^2)$ and $T_P \in O(\ell)$ for these compositions. In the following, we show that it suffices to compute the cost functions for a linear number of bends, leading to running times $T_s \in O(n^2)$ and $T_P \in O(n)$. Together with the time stated by Theorem 3, this gives us a total running time of $O(n^4)$.

Before we show the statement about the bends, we first establish a basic fact from flow theory.

Claim. *Let $N = (V_N, A_N)$ be a min cost flow network with multiple sources and sinks and non-decreasing cost functions, and let d_N denote the sum of all positive demands (i.e., the demand of all sinks in the network). Then there exists an optimal flow φ satisfying all capacities and demands such that the outflow of each vertex is at most d_N.*

Consider a minimum cost flow φ satisfying the demands and capacities. Suppose that a vertex v has outflow more than d_N, i.e., there is more flow leaving v than the total demand of all sinks in the network. It follows that at least one unit of flow is routed along a cycle C through N back to v. Reducing the flow along C yields a new valid flow with at most the same cost. We repeat this until, eventually, all vertices have outflow at most d_N. This proves the claim.

Let G be an st-graph with non-decreasing cost functions assigned to the edges. We show the existence of an optimal orthogonal representation of G such that every pertinent graph, and also the root edge st has $O(n)$ bends. To this end, consider the flow network N from [5] for OptimalFlexDraw, which is very similar to Tamassia's flow network [6], but uses flow to represent rotation instead of angles and bends. This network has the property that the rotation of edges or vertices in faces is in one-to-one correspondence to flow between nodes representing these edges, vertices, and faces, respectively (incoming and outgoing flow in a face represent positive and negative rotation in this face, respectively). Moreover, the total demand of the sinks d_N is in $O(n)$. Let φ be an optimal solution of N where each vertex has outflow at most d_N, which exists by the above claim. Let $\mathcal{R}$ be the corresponding orthogonal representation and assume that a pertinent graph of H has at least $d_N + 1$ bends. The case that the root edge st has $d_N + 1$ bends is handled analogously. By definition, there is a path π bounding H with $\text{rot}(\pi) \leq -(d_N + 1)$. Let f denote the face of G that is incident to π, where the rotation of π is at most $-(d_N + 1)$. Then, in the flow network, there are at least $d_N + 1$ units that flow from f into H via the edges dual to π, i.e., f has an outflow of at least $d_N + 1$. This contradicts the choice of the flow.

Thus, we can restrict our search to orthogonal representations in which each split component has only up to d_N bends. This can be done by implicitly setting the costs to ∞ for larger values than d_N. This concludes the proof, as $d_N \in O(n)$ holds. □

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 the instance G of FlexDraw 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). More precisely, 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 first study cost functions of k-critical st-graphs. Afterwards, we show how to use the insights we got to give an FPT-algorithm for k-critical instances of FlexDraw. Note that the cost functions we consider in this section only attain the values 0 and ∞, indicating whether or not a valid representation with the given number of bends exists.

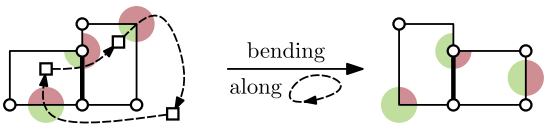

Fig. 5. An orthogonal representation (the bold edge is inflexible, other edges have flexibility 1), together with a valid cycle (dashed). Bending along this cycle increases the green and decreases the red angles. The resulting orthogonal representation is shown on the right. (For interpretation of the colors in this figure, the reader is referred to the web version of this article.)

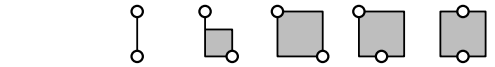

Fig. 6. Illustration of Fact 1 for some values of σ and τ.

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, that is 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. 5. 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 $\text{rot}(v_f) < 1$.

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.*

Proof. First, we show that the resulting rotations still describe an orthogonal representation. Afterwards, we show that this orthogonal representation is also valid and that it is a (σ, τ)-orthogonal representation. Let $e^\star = (g, f)$ be an edge in $C^\star$ with primal edge e. If Condition (I) holds, then $\text{rot}(e_g)$ is decreased by 1 and $\text{rot}(e_f)$ is increased by 1 and thus $\text{rot}(e_g) = -\text{rot}(e_f)$ remains true. Otherwise, Condition (II) holds and thus $\text{rot}(v_g)$ is reduced by 1 and $\text{rot}(v_f)$ is increased by 1. Obviously, the total rotation around v does not change. Moreover, both rotations remain in the interval $[-1, 1]$. Finally, the incoming arc to a face f in $C^\star$ increases the rotation around f by 1 and the outgoing arc decreases it by 1. Thus, the total rotation around each face remains as it was.

It remains to show that the resulting orthogonal representation is a valid (σ, τ)-orthogonal representation. First, Condition (I) ensures that we never increase the number of bends of an edge e above $\text{flex}(e)$. Moreover, due to the exception in Condition (II) where v is one of the poles, we never change the rotation of one of the poles. Thus the number of free incidences to the outer face does not change. □

As mentioned in Section 4, depending on σ and τ, there is a lower bound on the number of bends of (σ, τ)-orthogonal representations. We denote this lower bound by β_{low}; see Fig. 6.

Fact 1. *A (σ, τ)-orthogonal representation has at least $\beta_{\text{low}} = \left\lceil \frac{\sigma + \tau}{2} \right\rceil - 1$ 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. 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 $-\text{rot}(\pi(t, s)) \geq \beta_{\text{low}} + k + 1$ holds, then there exists a valid cycle $C^\star$ such that bending $\mathcal{R}$ along $C^\star$ reduces $-\text{rot}(\pi(t, s))$ by 1.*

Proof. We show the existence of a valid cycle $C^\star$ such that s and t lie to the left and right of $C^\star$, respectively. Obviously, such a cycle must contain the outer face. The edge in $C^\star$ having the outer face as target ensures that the rotation of an edge or a vertex of $\pi(t,s)$ is increased by 1 (which is the same as reducing $-\operatorname{rot}(\pi(t,s))$ by 1), where this vertex is neither s nor t (due to the exception of Condition (II)). Thus, $\operatorname{rot}(\pi(t,s))$ is increased by 1 when bending along $C^\star$ and thus $C^\star$ is the desired cycle. We first show the following claim.

Claim 1. *There exists a valid edge $e^\star$ that either has the outer face as source and corresponds to a primal edge e on the path $\pi(s,t)$, or is a loop with s to its left and t to its right.*

Assume the claimed edge $e^\star$ does not exist. We first show that the following inequality follows from this assumption. Afterwards, we show that this leads to a contradiction to the inequality in the statement of the lemma.

$$\operatorname{rot}(\pi(s,t)) \leq \begin{cases} k, & \text{if } \deg(s)=\deg(t)=1 \\ k-1, & \text{otherwise} \end{cases} \tag{2}$$

We first show this inequality for the case where we have **no critical and no semi-critical edges, in particular** $k=0$. We consider the rotation of edges and vertices on $\pi(s,t)$ in the outer face g. If an edge or vertex has two incidences to g, we implicitly consider the incidence corresponding to $\pi(s,t)$. Recall that the rotation along $\pi(s,t)$ is the sum over the rotations of its edges and of its internal vertices. The rotation of every edge e is $\operatorname{rot}(e_g) = -\operatorname{flex}(e)$ as otherwise $e^\star=(g,f)$ would be a valid edge due to Condition (I). At an internal vertex v we obviously have $\operatorname{rot}(v_g)\leq 1$, as larger rotations are not possible at vertices. Hence, as the flexibility of every edge is at least 1 and a path of ℓ edges has only $\ell-1$ internal vertices, we get $\operatorname{rot}(\pi(s,t))\leq -1$ and thus Equation (2) is satisfied.

Next, we allow **semi-critical edges, but no critical edges ($k=0$ remains)**. If $\pi(s,t)$ contains a semi-critical edge, it has a rotation of 0 (instead of -1 for normal edges). Note that we still assume that there is no critical edge in $\pi(s,t)$, i.e., $k=0$. Moreover, if an internal vertex v is incident to a semi-critical edge, it cannot have degree 4. In this case, there must be a face incident to v such that v has rotation at most 0 in this face. If this face was not g, Condition (II) would be satisfied. Thus, $\operatorname{rot}(v_g)\leq 0$ follows for this case. Consider the decomposition of $\pi(s,t)$ into maximal subpaths, each consisting of either only semi-critical edges or only of normal edges. It follows that each subpath consisting of semi-critical and normal edges has rotation at most 0 and -1, respectively. Moreover, the rotation at vertices between two subpaths is 0. Hence, if $\pi(s,t)$ contains at least one edge that is not semi-critical, we again get $\operatorname{rot}(\pi(s,t))\leq -1$ and thus Equation (2) is satisfied. On the other hand, if $\pi(s,t)$ consists of semi-critical edges, we get the weaker inequality $\operatorname{rot}(\pi(s,t))\leq 0$. If $\deg(s)=\deg(t)=1$ holds, Equation (2) is still satisfied as we have to show a weaker inequality in this case. Otherwise, one of the poles has degree at least 2 and thus the edges incident to it cannot be semi-critical by definition. Thus, the path $\pi(s,t)$ cannot consist of semi-critical edges.

Finally, we allow **critical edges, i.e.,** $k\geq 0$. If $\pi(s,t)$ contains critical edges, we first consider a hypothetical representation of $\pi(s,t)$, in which we replace each critical edge by an edge with flexibility 1 that has a left bend in the direction of $\pi(s,t)$, i.e., it contributes a rotation of -1 to $\operatorname{rot}(\pi(s,t))$. This does not create a valid edge $e^\star$. Hence, from the previous case, we obtain Equation (2) with $k=0$. To reobtain the original representation, we undo the replacement of the critical edges one by one. Each time $\operatorname{rot}(\pi(s,t))$ increases by 1. As $\pi(s,t)$ contains at most k critical edges, $\operatorname{rot}(\pi(s,t))$ is increased by at most k yielding Equation (2). This concludes the proof of Equation (2). Now we finish the proof of Claim 1.

In the case that $\deg(s)=\deg(t)=1$, the equation $\operatorname{rot}(\pi(s,t)) = -\operatorname{rot}(\pi(t,s))$ holds. Equation (2) together with the inequality in the statement of the lemma leads to $k\geq \beta_{\text{low}}+k+1$, which is a contradiction. In the following, we only consider the case where $\deg(s)=\deg(t)=1$ does not hold. Since the total rotation around the outer face sums up to -4, we get the following equation.

$$\operatorname{rot}(\pi(s,t)) + \operatorname{rot}(\pi(t,s)) + \operatorname{rot}(s_g) + \operatorname{rot}(t_g) = -4$$

Recall that $\operatorname{rot}(s_g)=\sigma-3$ and $\operatorname{rot}(t_g)=\tau-3$. Using Equation (2) ($\deg(s)=\deg(t)=1$ does not hold) and the inequality given in the lemmas precondition, we obtain the following.

$$\begin{aligned} & \Big(k-1\Big) - \Big(\overbrace{\left\lceil \frac{\sigma+\tau}{2}\right\rceil - 1}^{\beta_{\text{low}}} + k + 1\Big) + \Big(\sigma-3\Big) + \Big(\tau-3\Big) \geq -4 \\ \Leftrightarrow\quad & -\left\lceil \frac{\sigma+\tau}{2}\right\rceil + (\sigma+\tau) \geq 3 \\ \Leftrightarrow\quad & \left\lfloor \frac{\sigma+\tau}{2}\right\rfloor \geq 3 \end{aligned} \tag{3}$$

Recall that $\sigma+\tau\leq 5$ is a requirement of the lemma. Thus, Equation (3) is a contradiction, which concludes the proof of Claim 1.

Claim 2. *The valid cycle $C^\star$ exists.*

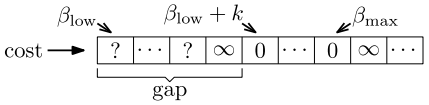

Fig. 7. A cost function with gap k.

Let $e^\star$ be the valid edge existing due to Claim 1. If $e^\star$ is a loop with s to its left and t to its right, then $C^\star = e^\star$ is the desired valid cycle. This case will serve as base case for a structural induction.

Let $e^\star = (g, f)$ be a valid edge dual to e having the outer face g as source. As $e^\star$ is not a loop, the graph $G - e$ is still connected and thus s and t are contained in the same block of the graph $G - e + st$. Let H be this block (without st) and let $\mathcal{S}$ be the orthogonal representation of H induced by $\mathcal{R}$. Then H is a k-critical st-graph, as H is a subgraph of G and $H + st$ is biconnected. Moreover, the path $\pi(t, s)$ is completely contained in H and thus its rotation does not change. Hence, all conditions for Lemma 6 are satisfied and since H contains fewer edges than G, we know by induction that there exists a valid cycle $C^\star_H$ such that bending $\mathcal{S}$ along $C^\star_H$ reduces $-\operatorname{rot}(\pi(t, s))$ by 1. As the dual graph $H^\star$ of H can be obtained from $G^\star$ by first contracting $e^\star$ and then taking a subgraph, all edges contained in $H^\star$ were already contained in $G^\star$. Moreover, all valid edges in $H^\star$ are also valid in $G^\star$ and thus each edge in $C^\star_H$ corresponds to a valid edge in $G^\star$. If these valid edges form a cycle in $G^\star$, then this is the desired cycle $C^\star$. Otherwise, one of the two edges in $C^\star_H$ incident to the outer face of H is in $G^\star$ incident to the outer face g of G and the other is incident to the face f of G. In this case the edges of $C^\star_H$ from in $G^\star$ a path from f to g and thus adding the edge $e^\star$ yields the cycle $C^\star$, which concludes the proof of Claim 2 and thus of this lemma. □

We get the following slightly weaker result for the case $\sigma = \tau = 3$.

Lemma 7. *Let G be a k-critical st-graph and let $\mathcal{R}$ be a valid $(3, 3)$-orthogonal representation. If $-\operatorname{rot}(\pi(t, s)) \geq \beta_{\text{low}} + k + 2$ 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.*

Proof. Since $\sigma = \tau = 3$ holds, we have $\beta_{\text{low}} = 2$ and thus $-\operatorname{rot}(\pi(t, s)) \geq k + 4$. We add an edge $e = ss'$ with flexibility 1 to G, where s' is a new vertex, and consider the orthogonal representation $\mathcal{R}'$ of $G + e$ where e has one bend such that e contributes a rotation of 1 to $\pi(t, s')$. Since the rotation at s in the outer face is 1, we have $\operatorname{rot}(\pi(t, s')) = \operatorname{rot}(\pi(t, s)) + 2$. If follows that $-\operatorname{rot}(\pi(t, s')) \geq k + 4 - 2 = k + 2$ holds. Since $\mathcal{R}'$ is a $(1, 3)$ orthogonal representation of $G + e$, and since the lower bound β'_{low} is 1 for $(1, 3)$ orthogonal representations, the precondition of Lemma 6, namely the inequality $-\operatorname{rot}(\pi(t, s')) \geq \beta'_{\text{low}} + k + 1$, is satisfied, which concludes the proof. □

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 $\mathcal{B}^\sigma_\tau$ 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, that is $\mathcal{B}^\sigma_\tau$ is not empty. We define the *maximum bend value* $\beta_{\max}$ to be the maximum in $\mathcal{B}^\sigma_\tau$. Moreover, let $\beta \in \mathcal{B}^\sigma_\tau$ be the smallest value, such that every integer between β and $\beta_{\max}$ is contained in $\mathcal{B}^\sigma_\tau$. Then we call the interval $[\beta_{\text{low}}, \beta - 1]$ the (σ, τ)-*gap* of G. The value $\beta - \beta_{\text{low}}$ is also called the (σ, τ)-*gap* of G; see Fig. 7.

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 $k + 1$.*

Proof. In the following, we assume $\sigma + \tau \leq 5$; the case $\sigma = \tau = 3$ works literally the same way when replacing Lemma 6 by Lemma 7. Let $\mathcal{R}$ be a valid (σ, τ)-orthogonal representation with $\beta \geq \beta_{\text{low}} + k + 1$ bends. We show the existence of a valid (σ, τ)-orthogonal representation with $\beta - 1$ bends. It follows that the number of bends can be reduced step by step down to $\beta_{\text{low}} + k$, which shows that the gap is at most k.

As $\mathcal{R}$ has β bends, either $-\operatorname{rot}(\pi(s, t)) = \beta$ or $-\operatorname{rot}(\pi(t, s)) = \beta$. Without loss of generality, we assume $-\operatorname{rot}(\pi(t, s)) = \beta \geq \beta_{\text{low}} + k + 1$. Due to Lemma 6 there exists a valid cycle $C^\star$, such that bending along $C^\star$ reduces $-\operatorname{rot}(\pi(t, s))$ by 1. This also reduces the number of bends by 1 (and thus yields the desired orthogonal representation) if $-\operatorname{rot}(\pi(s, t))$ is not increased above $\beta - 1$. Assume for a contradiction that $-\operatorname{rot}(\pi(s, t))$ was increased above $\beta - 1$. Then in the resulting orthogonal representation $-\operatorname{rot}(\pi(s, t))$ is greater than β_{low} and $-\operatorname{rot}(\pi(t, s))$ is at least β_{low}. It follows, that every (σ, τ)-orthogonal representation has more than β_{low} bends, which contradicts the fact that β_{low} is a tight lower bound. □

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.*

Proof. Recall that an orthogonal representation of G has β bends if either $-\operatorname{rot}(\pi(s,t)) = \beta$ or $-\operatorname{rot}(\pi(t,s)) = \beta$. We first consider the case that $-\operatorname{rot}(\pi(s,t)) = \beta$ for any number of bends $\beta \in [\beta_{\text{low}}, \beta_{\text{max}}]$.

By the definition of the gap, there exists a valid orthogonal representation for all choices of $-\operatorname{rot}(\pi(s,t)) \in [\beta_{\text{low}} + k, \beta_{\text{max}}]$, which forms the first interval. Moreover, G does not admit a valid orthogonal representation with $\beta_{\text{low}} + k - 1$ bends, since the gap would be smaller otherwise. Thus it remains to cover all allowed values contained in $[\beta_{\text{low}}, \beta_{\text{low}} + k - 2]$ by intervals. In the worst case, exactly every second value is possible. As $[\beta_{\text{low}}, \beta_{\text{low}} + k - 2]$ contains $k - 1$ integers, this results in $\lceil (k-1)/2 \rceil$ intervals of size 1. Thus, we can cover all allowed values for $\operatorname{rot}(\pi(s,t))$ in case $-\operatorname{rot}(\pi(s,t)) \in [\beta_{\text{low}} + k, \beta_{\text{max}}]$ holds using only $\lceil (k-1)/2 \rceil + 1$ intervals.

It remains to consider the case where G has β bends since $-\operatorname{rot}(\pi(t,s)) = \beta$ holds. With the same argument we can cover all possible values of $\pi(t,s)$ using $\lceil (k-1)/2 \rceil + 1$ intervals. As $\operatorname{rot}(\pi(s,t))$ equals $-\operatorname{rot}(\pi(t,s))$ shifted by some constant, we can cover all allowed values for $\operatorname{rot}(\pi(s,t))$ using $2 \cdot \lceil (k-1)/2 \rceil + 2$ intervals. If $k-1$ is even, this evaluates to $k+1$ yielding the statement of the lemma. If $k-1$ is odd and we assume the above-described worst case, then we need one additional interval. However, in this case there must exist a valid orthogonal representation with β_{low} bends and we counted two intervals for this bend number, namely for the case $-\operatorname{rot}(\pi(s,t)) = \beta_{\text{low}}$ and $-\operatorname{rot}(\pi(t,s)) = \beta_{\text{low}}$. We show that a single interval suffices to cover both cases by showing that either $-\operatorname{rot}(\pi(s,t)) = \beta_{\text{low}}$ or $-\operatorname{rot}(\pi(s,t)) = \beta_{\text{low}} - 1$ holds if $-\operatorname{rot}(\pi(t,s)) = \beta_{\text{low}}$. This again leads to the desired $k+1$ intervals.

Since the rotation around the outer face is -4, the equation $-\operatorname{rot}(\pi(s,t)) = \sigma + \tau - 2 + \operatorname{rot}(\pi(t,s))$ holds. For $-\operatorname{rot}(\pi(t,s)) = \beta_{\text{low}}$ we get the following.

$$\sigma + \tau - 2 - \beta_{\text{low}} = \sigma + \tau - 2 - \left\lceil \frac{\sigma + \tau}{2} \right\rceil + 1 = \left\lfloor \frac{\sigma + \tau}{2} \right\rfloor - 1$$

If $\sigma + \tau$ is even, this is equal to β_{low}, otherwise it is equal to $\beta_{\text{low}} - 1$, which concludes the proof. □

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 [6] 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*. Before we describe the demands of the nodes and the capacities of the arcs, we first explain the relation between orthogonal representations of G and flows in the network N. 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. Moreover, 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. To date, the best known upper bound for $T_{\text{flow}}(\ell)$ is $O(\ell \log^3 \ell)$ [11].

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. First note that in case of a rigid composition, computing "restricted" cost functions makes only a difference for the poles of the skeleton (as all other vertices have degree at least 3). However, enforcing 90° angles for the poles is already covered by the number of incidences the resulting graph occupies at its poles.

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. Before we show how to compute orthogonal representations of G, we show that the number of incidences σ_i and τ_i

a subgraph G_i occupies at its poles s_i and t_i is (almost) fixed. Assume that s_i is not one of the poles s or t of G. Then s_i has at least three incident edges in the skeleton H as $H + st$ is triconnected. Thus, the subgraph G_i occupies at most two incidences in any orthogonal representation of G, and hence s_i has either degree 1 or degree 2 in G_i. In the former case σ_i is 1, in the latter σ_i has to be 2. If s_i is one of the poles of G, then it may happen that G_i occupies two incidences in some orthogonal representations of G and three incidences in another orthogonal representation. However, this results in a constant number of combinations and thus we can assume that the values σ_i and τ_i are fixed for $i \in \{1, \dots, \ell\}$.

To test whether G admits a valid (σ, τ)-orthogonal representation, we can instead check the existence of a valid orthogonal representation of H using thick edges for the graphs $G_1, \dots, G_\ell$ (more precisely, we use a (σ_i, τ_i)-edge for G_i). To ensure that substituting the thick edges with the subgraphs yields the desired orthogonal representation, we have to enforce the following properties for the orthogonal representation of H. First, the orthogonal representation of H has to occupy σ and τ incidences at its poles. Second, the thick edge corresponding to a subgraph G_i is allowed to have β_i bends only if G_i has a valid (σ_i, τ_i)-orthogonal representation with β_i bends. Note that this tests the existence of an orthogonal representation without restriction to the number of bends. We will show later, how to really compute the cost function of G.

Restricting the allowed flows in the flow network such that they only represent (σ, τ)-orthogonal representations is easy. The graph H occupies σ incidences if and only if $\text{rot}(s_f) = \sigma - 3$ (where f is the outer face). As the rotation $\text{rot}(s_f)$ is represented by the flow on the corresponding vertex-face arc, we can enforce $\text{rot}(s_f) = \sigma - 3$ by setting the upper and lower bound on the corresponding arc to $\sigma - 3$. Analogously, we can ensure that H occupies τ incidences of t.

In the following we show how to restrict the number of bends of a thick edge $e_i = s_i t_i$ to the possible number of bends of the subgraph G_i it represents. Assume that G_i is k_i-critical. It follows from Lemma 8 that G_i has gap at most k_i. Thus, the possible values for $\text{rot}(\pi(s_i, t_i))$ can be expressed as the union of at most $k_i + 1$ intervals due to Lemma 9. Restricting the rotation to an interval can be easily done using capacities. However, we get $k_i + 1$ possibilities to set these capacities, and thus combining these possibilities for all thick edges results in $\prod(k_i + 1)$ flow networks.

We show that $\prod(k_i + 1)$ is in $O(2^k)$. To this end, we first show that $\sum k_i \leq k$ holds, by proving that an edge that is critical in one of the subgraphs G_i is still critical in the graph G. This is obviously true for critical edges in G_i not incident to a pole of G_i, as these inflexible edges already have endpoints with degree 4 in G_i. An edge e incident to a pole, without loss of generality s_i of G_i is critical in G_i if s_i has degree at least 2. If s_i remains a pole of G, then e is also critical with respect to G. Otherwise, s_i has degree 4 in G, which comes from the fact that the skeleton H becomes triconnected the edge st is added.

As the 0-critical subgraphs do not play a role in the product $\prod(k_i + 1)$, we only consider the d subgraphs $G_1, \dots, G_d$ such that G_i (for $i \in \{1, \dots, d\}$) is k_i-critical with $k_i \geq 1$. To find the worst case, we want to maximize $\prod(k_i + 1)$ with respect to $\sum k_i \leq k$ (which is equivalent to finding a hypercuboid of dimension d with maximal volume and with fixed perimeter). We get the maximum by setting $k_i = k/d$ for all subgraphs, which results in $(k/d + 1)^d$ combinations. Substituting $k/d = x$ leads to $x^{k/x}$, which becomes maximal, when $x^{1/x}$ is maximal. Since $f(x) = x^{1/x}$ is a decreasing function, we get the worst case for $x = 1$ (when restricting x to positive integers), which corresponds to $d = k$ graphs that are 1-critical. Thus, in the worst case, we get $O(2^k)$ different combinations.

Since the flow networks have size $O(\ell)$, we can test the existence of a valid orthogonal representation of G in $O(2^k \cdot T_{\text{flow}}(\ell))$ time. However, we want to compute the cost function instead. Assume we want to test the existence of a valid orthogonal representation with a fixed number of bends β. In the following, we show how to restrict each of the flow networks to allow only flows corresponding to orthogonal representation with β bends. Then G clearly admits a valid orthogonal representation with β bends if and only if one of these flow networks admits a valid flow. The orthogonal representation of H (and thus the resulting one of G) has β bends if either $-\text{rot}(\pi(s, t)) = \beta$ or $-\text{rot}(\pi(t, s)) = \beta$. We can consider these two cases separately, resulting in a constant factor in the running time. Thus, it remains to ensure that $-\text{rot}(\pi(s, t))$ is fixed to β. This can be done by splitting the face node corresponding to the outer face such that exactly the arcs entering f from edge nodes or vertex nodes corresponding to edges and internal vertices of $\pi(s, t)$ are incident to one of the resulting nodes. Restricting the flow between the two resulting nodes representing the outer face f to β obviously enforces that $-\text{rot}(\pi(s, t)) = \beta$ holds. Thus, we could get the cost function of G by doing this for all possible values of β. However, we can get the cost function more efficiently.

Instead of fixing the value of $-\text{rot}(\pi(s, t))$ to β, we can compute maximum flows to minimize or maximize it. Let $\text{rot}_{\min}$ and $\text{rot}_{\max}$ be the resulting minimum and maximum for $-\text{rot}(\pi(s, t))$, respectively. Recall that the number of bends of an orthogonal representation with $-\text{rot}(\pi(s, t)) < \beta_{\text{low}}$ is not determined by $\pi(s, t)$ but by the opposite path $\pi(t, s)$. Thus, if $\text{rot}_{\max}$ is less than β_{low}, then there is no orthogonal representation where the number of bends are determined by the rotation along $\pi(s, t)$. Moreover, if $\text{rot}_{\min} < \beta_{\text{low}}$, we set $\text{rot}_{\min} = \beta_{\text{low}}$. It follows from basic flow theory that all values between $\text{rot}_{\min}$ and $\text{rot}_{\max}$ are also possible. Thus, after computing the two flows, we can simply set the cost function of G to 0 on that interval. To save a factor of k in the running time we do not update the cost function of G immediately, but store the interval $[\text{rot}_{\min}, \text{rot}_{\max}]$.

In the end, we have $O(2^k)$ such intervals where the cost function is 0. The maximum of all upper bounds of these intervals is clearly $\beta_{\max}$ (the largest possible number of bends of G). It remains to extract the cost function of G on the interval $[\beta_{\text{low}}, \beta_{\text{low}} + k - 1]$, since the cost function of G has gap at most k (Lemma 8). This can be done by sorting all intervals having their lower bound in $[\beta_{\text{low}}, \beta_{\text{low}} + k - 1]$ by their lower bound. This can be done in $O(k + 2^k)$ time, since we sort $O(2^k)$ values in a range of size k. Finally, the cost function on $[\beta_{\text{low}}, \beta_{\text{low}} + k - 1]$ can be easily computed in

$O(k+2^k)$ time by scanning over this list. As this is dominated by the computation of all flows, we get an overall 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.*

Proof. First, consider only the non-restricted case. Let G_1 and G_2 be the two graphs that should be composed and let G be the resulting graph. As in the rigid case, we can use flow networks to compute the cost functions of G. However, this time the flow network has constant size and thus we do not have to be so careful with the constants.

Assume G_1 and G_2 are k_1- and k_2-critical, respectively. Up to possibly a constant number, all critical edges in G_i are also critical in G, that is $k_i \in O(k+1)$ (note that the "+1" is necessary for the case $k=0$). Thus, both graphs G_1 and G_2 have a gap of size $O(k+1)$. It follows that the possible rotations values for $\pi(s_i,t_i)$ (where s_i and t_i are the poles of G_i) are the union of $O(k+1)$ intervals, which results in $O(k^2+1)$ possible combinations and thus $O(k^2+1)$ flow networks of constant size. Note that we get an additional constant factor by considering all possible values for the number of occupied incidences of the graphs G_i. Extracting the cost functions out of the results from the flow computation can be done analogously to the case where we had a rigid composition (proof of Lemma 10), which finally results in the claimed running time $O(k^2+1)$.

To compute the restricted cost functions, one possibly has to restrict the rotation at some vertices to -1 or 1, which can be obviously done without increasing the running time. □

Theorem 5. FlexDraw *for k-critical graphs can be solved in $O(2^k \cdot n \cdot T_{\text{flow}}(n))$.*

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). 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 (that is we have k inflexible 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 [5] 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.

Acknowledgements

We thank Marcus Krug for discussions on FlexDraw. Moreover, we thank the anonymous reviewers for their useful comments.

References

[1] A. Garg, R. Tamassia, On the computational complexity of upward and rectilinear planarity testing, SIAM J. Comput. 31 (2) (2001) 601–625.
[2] T. Bläsius, M. Krug, I. Rutter, D. Wagner, Orthogonal graph drawing with flexibility constraints, Algorithmica 68 (4).
[3] T. Biedl, G. Kant, A better heuristic for orthogonal graph drawings, Comput. Geom. 9 (3) (1998) 159–180.
[4] G. Di Battista, G. Liotta, F. Vargiu, Spirality and optimal orthogonal drawings, SIAM J. Comput. 27 (6) (1998) 1764–1811.
[5] T. Bläsius, I. Rutter, D. Wagner, Optimal orthogonal graph drawing with convex bend costs, in: F.V. Fomin, R. Freivalds, M. Kwiatkowsak, D. Peleg (Eds.), Proceedings of the 40th International Colloquium on Automata, Languages and Programming, ICALP'13, in: Lecture Notes in Computer Science, vol. 7965, Springer, Berlin/Heidelberg, 2013, pp. 184–195.
[6] R. Tamassia, On embedding a graph in the grid with the minimum number of bends, SIAM J. Comput. 16 (3) (1987) 421–444.
[7] S. Cornelsen, A. Karrenbauer, Accelerated bend minimization, J. Graph Algorithms Appl. 16 (3) (2012) 635–650.
[8] G. Di Battista, R. Tamassia, On-line maintenance of triconnected components with SPQR-trees, Algorithmica 15 (4) (1996) 302–318.
[9] G. Di Battista, R. Tamassia, On-line planarity testing, SIAM J. Comput. 25 (5) (1996) 956–997.
[10] C. Gutwenger, P. Mutzel, A linear time implementation of SPQR-trees, in: Proceedings of the 8th International Symposium on Graph Drawing, GD'00, in: Lecture Notes in Computer Science, vol. 1984, Springer, Berlin/Heidelberg, 2001, pp. 77–90.
[11] G. Borradaile, P. Klein, S. Mozes, Y. Nussbaum, C. Wulff-Nilsen, Multiple-source multiple-sink maximum flow in directed planar graphs in near-linear time, in: Proceedings of the 52nd Annual Symposium on Foundations of Computer Science, FOCS'11, 2011, pp. 170–179.