The Effect of Almost-Empty Faces on Planar Kandinsky Drawings

Michael A. Bekos[1], Michael Kaufmann[1], Robert Krug[1], Martin Siebenhaller[2]

[1] Wilhelm-Schickard-Institut für Informatik, Universität Tübingen, Germany
{bekos,mk,krug}@informatik.uni-tuebingen.de
[2] yWorks GmbH, Tübingen, Germany
martin.siebenhaller@yworks.com

Abstract. Inspired by the recently-introduced slanted orthogonal graph drawing model, we introduce and study planar Kandinsky drawings with *almost-empty faces* (i.e., faces that were forbidden in the classical Kandinsky model [13]).
Based on the recent NP-completeness result for Kandinsky drawings by Bläsius et al. [4] we present and experimentally evaluate (i) an integer linear program that computes bend-optimal Kandinsky drawings with almost-empty faces, and, (ii) a more efficient heuristic that results in drawings with relatively few bends. Our experimental evaluation shows that the new model, in particular in the presence of many triangular faces, not only improves the number of bends, but also the compactness of the resulting drawings.

1 Introduction

The Kandinsky model [13] is a well-established graph drawing model that is heavily used in real applications; its roots date back to VLSI design and floor-planning applications. As a drawing model, it is a special type of grid embeddings [15,17], which, however, can be employed to draw any graph (that is, of arbitrary vertex-degree) in an orthogonal style. In this model, two grids are present; a coarse one to accommodate the vertices and a fine one to route the edges. More precisely, a Kandinsky drawing $\Gamma(G)$ of a graph G is one in which (a) every vertex is drawn as a box centered at a point of the underlying coarse grid, (b) all vertex boxes are of uniform size, (c) every edge is drawn as a sequence of alternating horizontal and vertical segments on the underlying fine edge-grid, and, (d) arbitrarily many edge-segments can be connected to each side of every vertex; see Fig. 1e.

Due to its high importance in practical applications, several different variants of this model have been proposed and studied over the years (see, e.g., [3,6,7,10,8]). The classical orthogonal model studied by Tamassia [16] can also be seen as a restricted variant of the Kandinsky drawing model, where the graphs have maximum degree four and no two edges can be attached to the same side of a vertex.

Beside the area minimization, a typical objective that has received considerable attention is the bend minimization. For plane graphs, i.e., planar graphs with a fixed planar embedding, this problem was initially modeled as a min-cost flow problem [13,8]. Subsequently, it has been observed that the given algorithm needed additional constraints that could not be handled efficiently, and therefore efficient 2-approximations have been proposed [1,9]. The complexity of the problem was unknown for more than two decades until the NP-completeness result was recently found [4]. As in previous work, we only consider plane graphs.

Most of the known algorithms [1,9,13] for the Kandinsky drawing model heavily depend on the absence of *empty faces*; see Fig. 1a and 1b. A common approach to avoid such faces

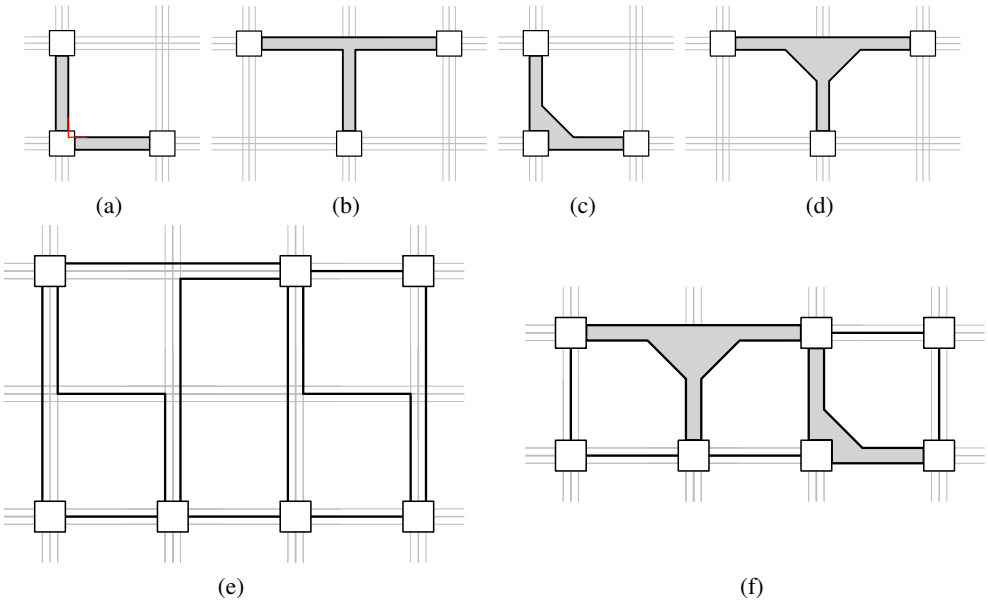


Fig. 1. (a)-(b) Illustration of empty faces: (a) empty L, and, (b) empty T; (c)-(d) Illustration of almost-empty faces: (c) almost-empty L, and, (d) almost-empty T; (e)-(f) Sample Kandinsky drawings: (e) without almost empty faces, and, (f) with almost empty faces, which results in height reduction.

is to adopt the so-called *bend-or-end property*. In the work of Fößmeier and Kaufmann [13], this property is the result of the min-cost flow formulation which assigns each angle of zero degrees between two edges to a specific bend on one of these edges. From a practical point of view, such faces are usually forbidden because it is too difficult to distinguish them in a drawing (as they are of almost zero area). In addition, the empty L (see Fig. 1a) is not possible to be drawn without introducing vertex-edge crossings.

Inspired by recent work of Bekos et al. [2] introducing the so-called slanted orthogonal graph drawing model, we use intermediate diagonal edge-segments (at 45° angles), which allow us to draw empty faces; see Fig. 1c and 1d. In this way, we can create "empty faces" of non-zero area, making them acceptable in a drawing. We refer to such faces as *almost-empty faces*. To maintain a uniform approach in the way we draw the bends of the edges in the drawing, we replace all 90° bends by pairs of *half-bends* of 135°; see Fig. 2a and 2b.

For aesthetic reasons, we further require that all diagonal edge-segments are short and of uniform length. In particular, we perform our comparisons based on the assumption that the length of a diagonal edge-segment does not exceed one third of the length of one unit of the underlying coarse grid in both x- and y-direction. This still allows us to draw a U-shaped edge in a 1×1 integer grid (as in the classical Kandinsky drawing); see Fig. 2c and 2d. For a sample drawing refer to Fig. 1f, where the graph of Fig. 1e has been redrawn in the new model. Also, observe that the height of the new drawing has been decreased by one unit.

We refer to such drawings as *Kandinsky drawings with almost-empty faces* (or *podevsaef-drawings*[3]). Formally, a podevsaef drawing $\Gamma(G)$ of a plane graph G is one in which:

[3] The term is inspired by a term that also refers to Kandinsky drawings: *podevsnef drawings* [13], which stands for Planar Orthogonal Drawings with Equal Vertex Sizes and No Empty Faces.

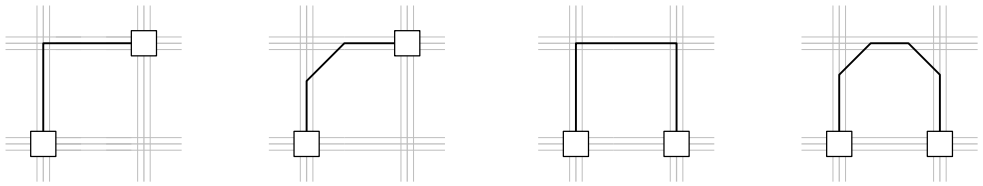


Fig. 2. (a)-(b) Replacing a 90° bend by a pair of half-bends of 135°; (c)-(d) Illustration of a U-shaped edge drawn in the classical orthogonal model and in the new model.

(a) every vertex is drawn as a box centered at a point of the underlying coarse grid,
(b) all vertex boxes are of uniform size,
(c) every edge is drawn as a sequence of alternating horizontal, vertical and diagonal segments on the underlying fine-grid,
(d) arbitrarily many edge-segments can be connected to one side of a vertex (so the resolution of the underlying fine-grid has to be high enough to accommodate all edges),
(e) a diagonal segment is of length at most one third of the length of one unit of the underlying coarse grid in both x- and y-direction and is never incident to a vertex,
(f) the minimum of the angles formed by two consecutive segments of an edge always is 135°, which suggests that a bend in $\Gamma(G)$ is always incident to a diagonal segment and to either a horizontal or a vertical one.

For sample podevsaef drawings produced by implementations of our algorithms refer to Fig. 3b and 3c; the corresponding bend-optimal Kandinsky drawing of the same graph is given in Fig. 3a.

Our goal is to find out how much we can save with respect to bends and area when allowing almost-empty faces in Kandinsky drawings. Note that almost-empty faces are always triangular. So, we expect that the improvements will be greater in graphs with many triangular faces. Since the recent NP-completeness result of Bläsius et al. [4] implies that also our problem is NP-complete, we take an experimental approach. To quantify the results of our experiments, we present an ILP-formulation, which extends the standard one of Eiglsperger et al. [11] and which results in bend-optimal podevsaef-drawings (see Section 2). By relaxing the bend-optimality constraint on the resulting drawings, we are able to present an efficient heuristic which results in podevsaef-drawings with relatively few bends (see Section 3). In Section 4 we experimentally evaluate the podevsaef drawing model and we also compare it with the classical Kandinsky drawing model. We conclude in Section 5 with open problems and future work.

2 Bend-Optimal Podevsaef Drawings of Planar Graphs

In this section, we present an approach that results in podevsaef drawings of minimum number of bends. Following standard practice, our approach consists of two phases; the *orthogonalization phase* (where the angles and the bends of the drawing are computed) and the *compaction phase* (where the actual coordinates for the vertices and the edges are computed); refer, e.g., to [16]. For the orthogonalization phase, we present a modification of a standard ILP-formulation of Eiglsperger et al. [11] that results in representations of minimum number

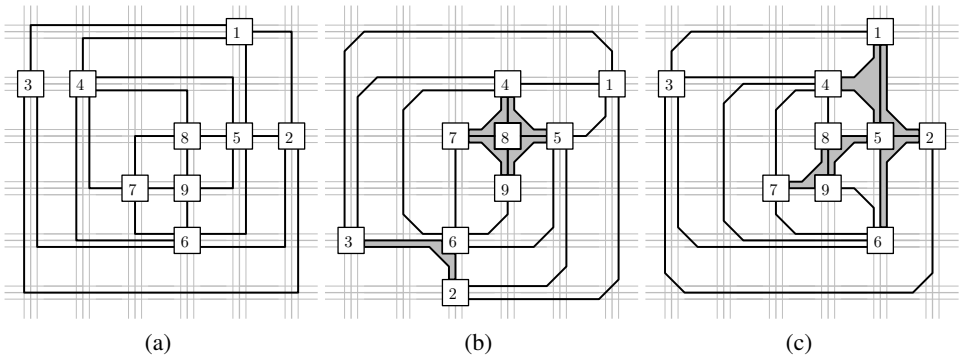


Fig. 3. A sample planar graph drawn (a) in the classical Kandinsky drawing model, (b) in the podevsaef drawing model using the ILP of Section 2, and, (c) in the podevsaef drawing model using the heuristic of Section 3. Almost-empty faces are drawn gray.

of bends. In the compaction phase, the computed representation is transformed into an actual drawing. To do so, we employ a simple transformation that allows us to use any known compaction algorithm for the original Kandinsky model.

The Orthogonalization Phase: Before we proceed with the description of our modification for the orthogonalization phase, we first quickly recall the ILP-formulation of Eiglsperger et al. [11]. For each edge $e = (u, v)$, variable $a_{(u,v)} \cdot 90^\circ$ corresponds to the angle formed by edge e and its cyclic predecessor at vertex u. Clearly, $a_{(u,v)} \in \{0, 1, 2, 3, 4\}$. Since the sum of the angles around a vertex equals to 360°, it follows that for each vertex $u \in V$, $\sum_{(u,v)\in N(u)} a_{(u,v)} = 4$ must hold, where $N(u)$ denotes the neighbors of u.

In order to count the number of left turns (or simply *left-bends*) along an edge $e = (u, v)$, three variables, $lb^u_{(u,v)}$, $lb^v_{(u,v)}$ and $lb_{(u,v)}$, are employed, which correspond to the left Kandinsky-bend (that is, the special bend resulting from the bend-or-end property) at vertex u, the left Kandinsky-bend at vertex v and the remaining left-bends of edge (u, v). For the right-bends, variables $rb^u_{(u,v)}$, $rb^v_{(u,v)}$ and $rb_{(u,v)}$ are defined similarly. Clearly, for reasons of symmetry $lb^u_{(u,v)} = rb^u_{(v,u)}$, $lb_{(u,v)} = rb_{(v,u)}$ and $lb^v_{(u,v)} = rb^v_{(v,u)}$ must hold. Note that variables $lb^u_{(u,v)}$, $lb^v_{(u,v)}$, $rb^u_{(u,v)}$ and $rb^v_{(u,v)}$ are binary, while variables $lb_{(u,v)}$ and $rb_{(u,v)}$ are non-negative integers.

Since only one Kandinsky-bend is allowed at each end of each edge, $lb^u_{(u,v)} + rb^u_{(u,v)} \leq 1$ must hold for each edge $(u, v) \in E$. For ease of notation, we denote by $l_{(u,v)}$ and $r_{(u,v)}$ the total number of left and right bends per edge, respectively, that is, $l_{(u,v)} = lb^u_{(u,v)} + lb_{(u,v)} + lb^v_{(u,v)}$ and $r_{(u,v)} = rb^u_{(u,v)} + rb_{(u,v)} + rb^v_{(u,v)}$. Since the sum of the angles formed at the vertices and at the bends of a bounded face f equals to $180 \cdot (p(f) - 2)$, where $p(f)$ denotes the number of such angles, it follows that $\sum_{(u,v)\in f}(a_{(u,v)} + l_{(u,v)} - r_{(u,v)}) = 2a(f) - 4$, where $a(f)$ denotes the number of vertex angles in f. If f is not bounded, the corresponding sum is increased by 8. Empty faces are forbidden by requiring $a_{(v,u)} + lb^v_{(v,w)} + rb^v_{(v,u)} \geq 1$, for all pairs of consecutive edges (v, w) and (v, u) around v. Of course, the objective function of the corresponding ILP-formulation must minimize the sum of all (i.e., either left or right) bends, that is $\min \sum_{(u,v)\in E}(l_{(u,v)} + r_{(u,v)})$. The complete program is given in Linear Program 1.

To enable the aforementioned ILP-formulation to use the almost-empty T and almost-empty L-shapes (see Fig. 1a and 1b, respectively), we observe that a bend of a classical

$$
\begin{array}{lll}
\textbf{min} \ \sum_{(u,v)\in E}(l_{(u,v)} + r_{(u,v)}) & & \\
\textbf{s.t.} \ a_{(u,v)} \in \{0,1,\ldots,4\} & \forall (u,v) \in E & (1)\\
\sum_{(u,v)\in N(u)} a_{(u,v)} = 4 & \forall u \in V & (2)\\
\sum_{(u,v)\in f}(a_{(u,v)} + l_{(u,v)} - r_{(u,v)}) & & \\
\quad = \begin{cases} 2a(f)-4; \ f \text{ bounded} \\ 2a(f)+4; \ f \text{ unbounded} \end{cases} & \forall f \in F & (3)\\
lb^u_{(u,v)} + rb^u_{(u,v)} \leq 1 & \forall (u,v) \in E & (4)\\
lb^u_{(u,v)} = rb^u_{(v,u)} & \forall (u,v) \in E & (5)\\
lb_{(u,v)} = rb_{(v,u)} & \forall (u,v) \in E & (6)\\
lb^v_{(u,v)} = rb^v_{(v,u)} & \forall (u,v) \in E & (7)\\
a_{(v,u)} + lb^v_{(v,w)} + rb^v_{(v,u)} \geq 1 & \forall (v,w),(v,u) \text{ subsequent in } N(v) & (8)\\
lb^u(u,v), rb^u(u,v) \in \{0,1\} & \forall (u,v) \in E & (9)\\
lb(u,v), rb(u,v) \in \mathbb{N} & \forall (u,v) \in E & (10)
\end{array}
$$

Linear Program 1: The ILP of Eiglsperger et al. [11] for computing bend-optimal Kandinsky representations.

Kandinsky drawing always corresponds to a pair of half-bends in our drawing model. So, in our formulation we are working with bends (not half-bends), which we eventually replace with pairs of half-bends only in the compaction phase. In addition, we replace Constraint 8 of Linear Program 1 with new constraints that we describe in the following (refer to Constraint Sets 1 and 2). For each triangular face f, we introduce two binary variables, say T_f and L_f, that are set to one if and only if f is drawn using the almost-empty T or the almost-empty L-shape, respectively. We also employ a large constant M which we use to "activate" or "deactivate" constraints; a common trick used in ILPs. It is known, however, that the choice of the value for the constant M might significantly influence the time required to compute an optimal solution of an ILP [5]. Our experimental evaluation showed, however, that in our case the choice of the value of M did not have any particular effect on the computation time.

If variable T_f of face f is set to one (that is, f is drawn using the almost-empty T-shape), then Constraint 12 ensures that all angles of face f are zero. Constraints 13 and 14 force face f to have in total two right-bends on all edges (hence, the third edge of face f must be bend-less). Constraint 15 ensures that no edge has a left-bend and Constraint 16 guaranteed that all edges have at most one bend in total. On the other hand, if T_f is set to zero, then these constraints are all deactivated, so that they impose no restriction on the edges.

The constraints for an L-shaped face f are similar. If variable L_f is set to one, Constraints 17 and 18 ensure that there is one $90°$ angle in f. Constraints 19 and 20 force f to have exactly one right-bend and Constraint 21 makes sure that there are no left-bends. Again, setting L_f to zero trivially fulfills all these constraints and they pose no restriction on f.

$$
\begin{array}{ll}
\sum_{e\in E_f} a_e \leq 0 + (1-T_f)\cdot M & (12)\\
\sum_{e\in E_f} r_e \geq 2 - (1-T_f)\cdot M & (13)\\
\sum_{e\in E_f} r_e \leq 2 + (1-T_f)\cdot M & (14)\\
\sum_{e\in E_f} l_e \leq 0 + (1-T_f)\cdot M & (15)\\
\forall e \in E_f : r_e \leq 1 + (1-T_f)\cdot M & (16)
\end{array}
$$

Constraint Set 1: Face f is T-shaped

$$
\begin{array}{ll}
\sum_{e\in E_f} a_e \leq 1 + (1-L_f)\cdot M & (17)\\
\sum_{e\in E_f} a_e \geq 1 - (1-L_f)\cdot M & (18)\\
\sum_{e\in E_f} r_e \leq 1 + (1-L_f)\cdot M & (19)\\
\sum_{e\in E_f} r_e \geq 1 - (1-L_f)\cdot M & (20)\\
\forall e \in E_f : l_e \leq 0 + (1-L_f)\cdot M & (21)
\end{array}
$$

Constraint Set 2: Face f is L-shaped

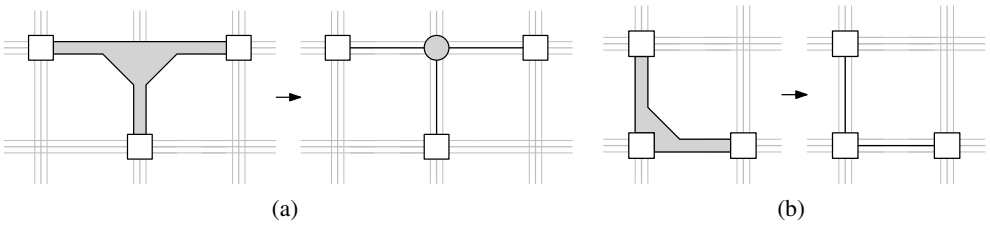


Fig. 4. (a) Transformation for T-shapes. (b) Transformation for L-shapes.

Observe that the constraints for T-shaped and L-shaped faces exclude each other. So, there is no reason to add an extra constraint for this purpose. Since we intend to allow either the almost-empty T or the almost-empty L-shape, it follows that it suffices to replace Constraint 8 of Linear Program 1 with the following constraint for each triangular face f:

$$a_{(u,v)} + lb^v_{(v,w)} + rb^v_{(v,u)} + T_f + L_f \geq 1,\ \forall (v,w),(v,u) \text{ subsequent in } N(v)$$

In order to prove that the modified ILP-formulation results in correct podevsaef representations, we observe that if we set all T_f and L_f variables to zero, then all new constraints no longer affect the underlying equation system and all old constraints stay exactly as before. So, the original proof of correctness of Eiglsperger et al. [11] holds. On the other hand, it is not difficult to see that if a face is to be drawn either as an almost-empty T or as an almost-empty L-shape, then Constraint Sets 1 and 2, respectively, ensure that all angles and bends are correctly computed. So, the podevsaef representation is correctly computed.

The Compaction Phase: As already stated, in the compaction phase, where the computed representation has to be transformed into an actual drawing (that is, the actual coordinates of the vertices and edges have to be computed), we employ a simple transformation that allows us to use any known algorithm for the compaction phase of the original Kandinsky model, e.g., [12]. The transformation is illustrated in Fig. 4. More precisely, for an almost-empty T-shaped face a new auxiliary vertex is required (refer to the gray colored vertex in Fig. 4a) and the angles around it follow directly from the T-shape. Similarly, for an almost-empty L-shaped face we simply ignore the bent edge involved (see Fig. 4b).

Once all almost-empty T-shaped and L-shaped faces are transformed according to the rules of Fig. 4, we proceed to draw the new graph using any known compaction algorithm for the original Kandinsky model. In the resulting drawing, the applied transformations can be easily reversed by introducing the missing edges of the L-shaped faces and replacing the auxiliary vertices of the T-shaped faces with the original edges.

3 A Heuristic to Compute Podevsaef Drawings

As we will shortly see in Section 4, the running time needed to compute bend-optimal podevsaef drawings may be high, because of the underlying integer linear program that is used to compute the corresponding bend-optimal podevsaef representation. So, in this section, we present a significantly more efficient heuristic which given an orthogonal representation of minimum number of bends, computes a podevsaef drawing with as few half-bends as possible. In addition, our experimental evaluation shows that the produced drawings are comparable to the optimal ones in terms of the total number of bends and the area requirement, which suggests that the proposed heuristic is definitely useful for practical applications.

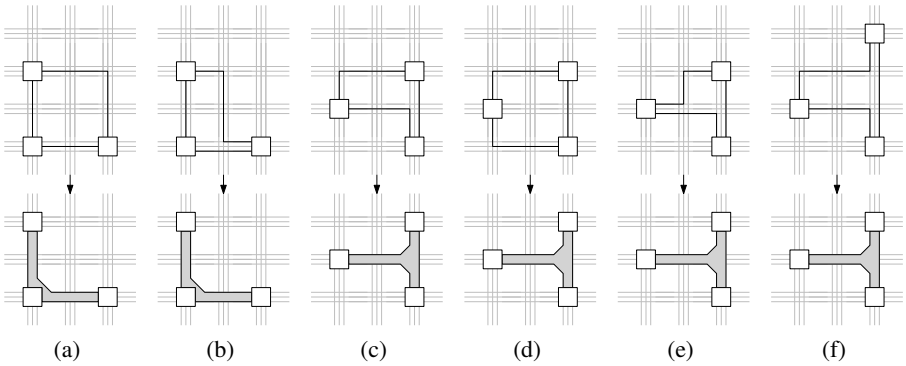


Fig. 5. (a),(b) Shapes that can be transformed into an L. (c)-(f) Shapes that can be transformed into a T.

The main idea of our approach is to start from a classical bend-optimal orthogonal representation and heuristically try to modify the shape of as many triangular faces as possible to become T or L-shaped. To achieve this, we have identified several shapes that allow an easy transformation into the new almost-empty shapes; see Fig. 5. As in the previous section, these transformations do not require a drawing, but they are directly applicable to a given orthogonal representation (which, in addition, is not required to be bend-optimal). Regarding the total number of bends, it is not difficult to see that for each transformation the number of half-bends of the podevsaef representation either equals to twice (see Fig. 5a, 5c, 5d) or is even less than twice (see Fig. 5b, 5e, 5f) the number of bends of the orthogonal representation.

Once all triangular faces have been transformed according to the rules of Fig. 5, we proceed with the compaction phase to obtain the final drawing (as described in the previous section). Alternatively, we could heuristically try to further reduce the number of bends by adding another orthogonalization step. This is because the transformed graph may allow a drawing with even less bends. However, particular attention must be payed on keeping the shape of the edges that have been transformed unchanged in subsequent steps. In particular, the angles around the grey colored vertex of Fig. 4a must not be changed and its incident edges must not be bent. Similarly, one copes with the almost-empty L shape of Fig. 4b.

In order to further improve the quality of the heuristic (in terms of the number of bends in the resulting layouts), we employ a preprocessing of the input, which according to our evaluation has proved to be very effective. We observe that when the input representation is computed in such a way that it is bend-optimal and simultaneously contains the maximum number of S-shaped edges, then the number of half-bends in the resulting podevsaef drawings tends to be reduced. This is because all transformations presented in Fig. 5 that involve S-shaped edges require less half-bends than twice the corresponding number of bends. To achieve this, we introduce appropriate binary variables, which determine whether an edge is S-shaped, and employ them to modify (or, more precisely, weight) the objective function of the integer linear program of Eiglsperger et al. [11] so to "prefer" an S-shaped edge rather than two edges with a single bend each, whenever this is possible.

4 An Experimental Evaluation of the Podevsaef Drawing Model

In this section, we present an experimental evaluation of the podevsaef drawing model. In particular, we compared bend-optimal Kandinsky drawings obtained by implementing the orig-

inal integer linear program of Eiglsperger et al. [11] with bend-optimal podevsaef drawings computed with the algorithm presented in Section 2 and close-to-optimal drawings computed by the heuristic presented in Section 3.

Experiment's setup: We implemented all aforementioned algorithms using Java and the yFiles library (http://www.yworks.com). The gurobi solver [14] was employed to solve the different linear programs. The experiment was performed on a desktop Linux machine with four cores at 2.5 GHz and 3 GB RAM.

As a test set for our experiment, we used three different graph-suites, each containing planar graphs of different densities (recall that the *density* of a graph is defined as the ratio of the number of its edges to the number of its vertices): (i) the *planar Rome graphs*, which form a collection of 3279 graphs with average density of 1.16 obtained from the graphdrawing.org website, (ii) the *planar North graphs*, which form a collection of 854 graphs with average density of 1.14 also obtained from the graphdrawing.org website, and, (iii) 940 randomly created triangulations with average density 2.82 which were created (based on the yFiles approach) as follows. Initially, an evenly distributed point set was created within a triangular region $\mathcal{T}$. Then, the points were sorted from left to right. The first three points formed a triangle and each following point was connected with the visible points to its left. Finally, the points that were still on the boundary of the created drawing were appropriately connected to three additional vertices that reside on the corners of $\mathcal{T}$.

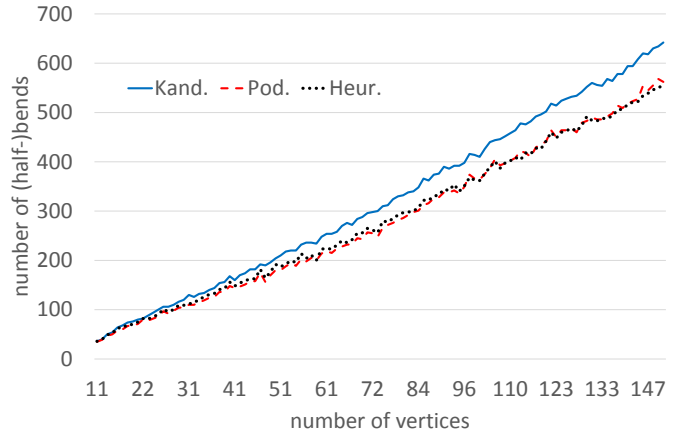
Due to space constraints, in the remainder of this section we present results only for the test set of the randomly created triangulations. These graphs contain only triangular faces. So, we expect that they will better show the effect of the almost-empty faces on the Kandinsky model. The results for the remaining test sets are given in the appendix.

From our experiment, we quickly realized that the time required to solve the ILP which are used to compute bend-optimal podevsaef representations increases rapidly with the number of triangular faces of the graph. So, we set a time-limit of 300 seconds in our experiment. If the solver was able to find a feasible solution within this time-limit, then the solution closest to the optimal one was used for our evaluation. Otherwise, the instance counted as failed and was excluded from the experiment. In total, we found just two such faulty instances, which both stemmed from the test set of the randomly created triangulations.

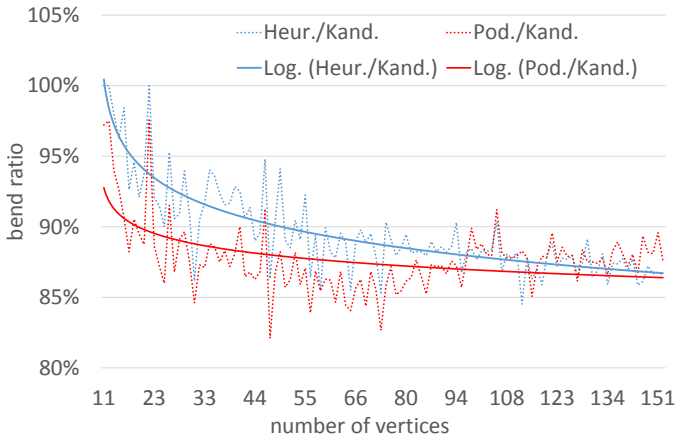
To obtain an input for our algorithms, we applied the combinatorial embedder of the yFiles graph library, which guarantees that if the input graph is planar, then the computed combinatorial embedding will be planar as well.

We are now ready to present the results of our experimental evaluation. In all following plots, the curve denoted by "Kand" stands for results for orthogonal drawings, while the curves denoted by "Pod" and "Heur" correspond to the results for bend-optimal and heuristically computed podevsaef drawings, respectively. Also, the values for a specific number of vertices were obtained by averaging over all instances with the same number of vertices.

Number of bends: In Fig. 6a the required number of bends is plotted against the number of vertices for the test set of the randomly generated triangulations. Since a bend of a classical Kandinsky drawing always corresponds to a pair of half-bends of a podevsaef drawing (which are necessary for the short intermediate diagonal edge-segments of our model), in Fig. 6a we plotted twice the number of orthogonal bends against the number of half-bends produced by our algorithms. As expected, the number of half-bends of bend-optimal (or, more precisely, close-to-bend-optimal) podevsaef drawings is significantly less than twice the number of bends of bend-optimal classical Kandinsky drawings, especially for graphs with relatively many vertices. As illustrated in Fig. 6b, the reduction of the number of bends



(a)



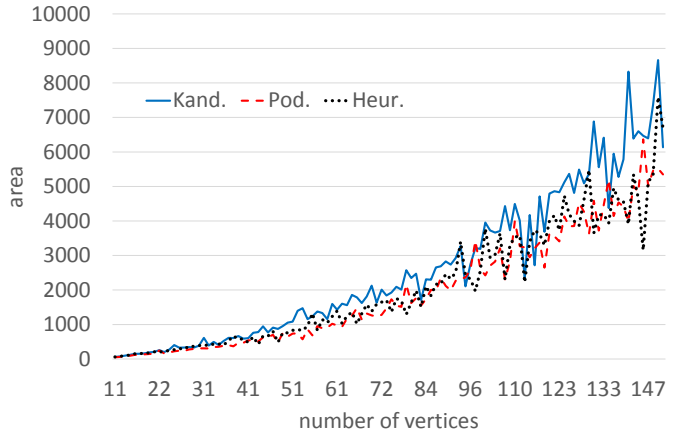
(b)

Fig. 6. Experimental results for the test set of the randomly created triangulations: (a) The total number of bends is plotted against the number of vertices, (b) the ratio and the logarithm of the ratio of the total number of bends of our algorithms to the total number of bends of bend-optimal Kandinsky drawings is plotted against the number of vertices.

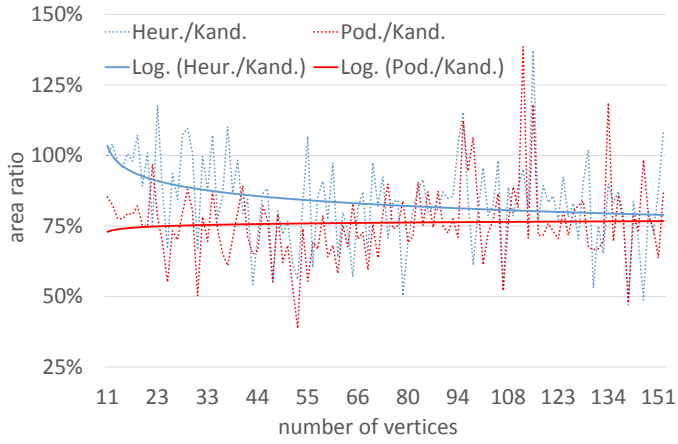
for both algorithms tends to be around 13% with respect to the classical Kandinsky drawings. The reason is that the graphs of our test set contain only triangular faces, which facilitates the bend-reduction under the podevsaef drawing model.

It is worth mentioning, though, that the drawings produced by the ILP of Section 2 and the ones produced by the heuristic of Section 3 are of comparable number of half-bends. More importantly, both seem to have the same tendency, as can be seen in Fig. 6b. This justifies our claim that the heuristic is of practical importance.

For the test sets of planar Rome and planar North graphs the profit is significantly smaller; see Fig. 8a and 9a in the appendix. More precisely, all algorithms seem to produce drawings with very similar number of bends on average. This is explained by the fact that both the planar Rome and the planar North graphs are very sparse and with very few triangular faces.



(a)



(b)

Fig. 7. Experimental results for the test set of the randomly created triangulations: (a) The area requirement is plotted against the number of vertices, (b) the ratio and the logarithm of the ratio of the area of our algorithms to the area of bend-optimal Kandinsky drawings is plotted against the number of vertices.

Area requirements: In Fig. 7a the required area is plotted against the number of vertices for the test set of the randomly generated triangulations. Again, for graphs with relatively many vertices the area required for podevsaef-drawings is less than the corresponding one for classical Kandinsky drawings. In addition, we observed that the ILP of Section 2 and the heuristic of Section 3 seem to have comparable performance in terms of area requirements, which according to Fig. 7b corresponds to an area reduction of around 20% with respect to the classical Kandinsky drawings.

It is also worth mentioning that in several cases the drawings computed by the heuristic of Section 3 were more compact than those produced by the ILP of Section 2. However, both seem to have the same tendency in terms of the area requirements, as can be seen in Fig. 7b.

For the other two test sets of our experiments, it seems that the podevsaef drawings (both the bend-optimal ones and the ones created by the heuristic of Section 3) and the classical Kandinsky ones require comparable area; see Fig. 8b and 9b in the appendix.

Running time: On the negative side, the time required by the ILP to compute bend-optimal podevsaef representations increases rapidly with the number of triangular faces of the graph. More precisely, in the test set of the randomly created triangulations, we observed that for graphs with more than 20 vertices the time required to compute an optimal podevsaef drawing exceeded the time-limit of 300 seconds. On the other hand, this negative behavior was difficult to be observed for the test sets of planar Rome and planar North graphs, where all instances could be solved within a few seconds (see Fig. 10 in the appendix). The reason is that these graphs are much more sparse and have very few triangular faces. This shows that the test set of the randomly created triangulations was the most demanding one in terms of running time.

On the positive side, however, it was almost always possible to compute at least a close-to-optimal solution within the time-limit we set (except for just two faulty cases, as we already mentioned). On the other hand, both the ILP of Eiglsperger et al. [11] and the heuristic of Section 3 seem to require comparable running times; in all test cases less than two seconds. Since the heuristic of Section 3 eventually produced drawings of comparable number of bends and area (with respect to the ones produced by the ILP of Section 2), it confirms our claim that it is more suitable for practical applications.

5 Conclusion and Open Problems

In this paper, we introduced and experimentally evaluated the podevsaef graph drawing model that is appropriate for drawing planar graphs of arbitrary vertex-degree. Since the problem of minimizing the total number of bends in this model turned out to be NP-complete, we modeled it as an ILP and we also presented a more efficient heuristic. Our evaluation showed that the suggested graph drawing model is able to improve the quality of a classical Kandinsky drawing (in terms of total number of bends and area requirements) in the presence of many triangular faces. We strongly believe that our new model is of particular importance, as the Kandinsky drawing model is well-established and widely used in practical applications. Of course, our work is ongoing and raises several open questions:

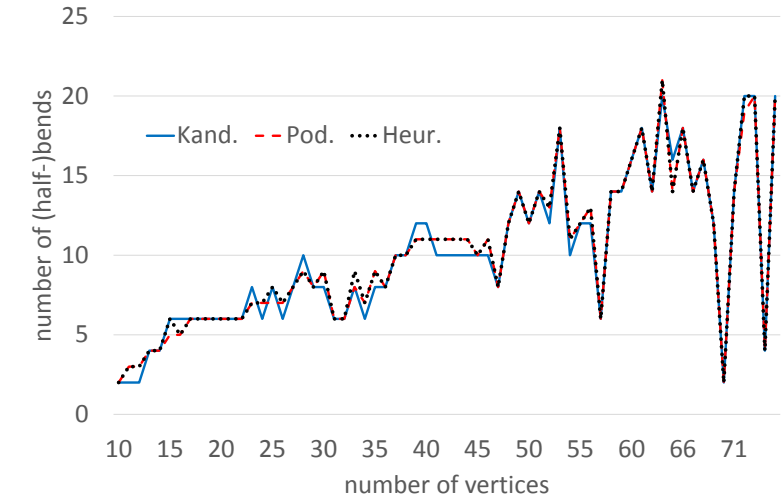
1. A more sophisticated heuristic or a constant-factor approximation algorithm for computing close-to-optimal podevsaef drawings would be of interest.
2. A different approach that will allow for faster computation of optimal (in terms of the total number of bends) podevsaef drawings, especially when the input graph is triangulated, is also of interest.
3. We only considered the bend-minimization problem. Is there an efficient algorithm that results in podevsaef drawings of provable small drawing area for given plane graphs, especially for triangulations? Are there any non-trivial bounds that one could derive?
4. We considered only planar graphs. It is of interest to extend the proposed model also to the case of non-planar graphs. A reasonable research direction is again to adopt the slanted orthogonal drawing model, which restricts all edge-crossings to diagonal edge-segments at 45°.

References

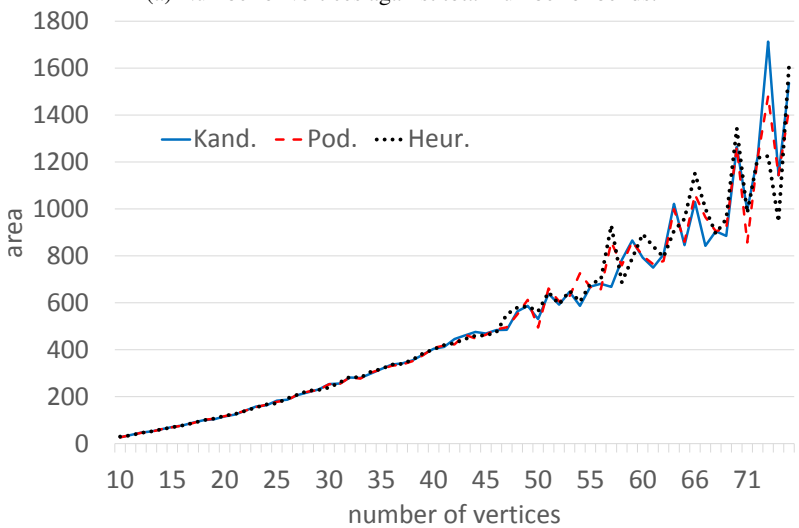
1. W. Barth, P. Mutzel, and C. Yildiz. A new approximation algorithm for bend minimization in the kandinsky model. In M. Kaufmann and D. Wagner, editors, *Graph Drawing*, volume 4372 of *Lecture Notes in Computer Science*, pages 343–354. Springer Berlin Heidelberg, 2007.
2. M. A. Bekos, M. Kaufmann, R. Krug, S. Näher, and V. Roselli. Slanted orthogonal drawings. In S. K. Wismath and A. Wolff, editors, *Graph Drawing*, volume 8242 of *LNCS*, pages 424–435. Springer, 2013.
3. P. Bertolazzi, G. Di Battista, and W. Didimo. Computing orthogonal drawings with the minimum number of bends. *Computers, IEEE Transactions on*, 49(8):826–840, Aug 2000.
4. T. Bläsius, G. Brückner, and I. Rutter. Complexity of higher-degree orthogonal graph embedding in the kandinsky model. In A. S. Schulz and D. Wagner, editors, *Symposium on Algorithms*, volume 8737 of *LNCS*, pages 161–172. Springer, 2014.
5. Y. D. Der-San Chen, Robert G. Batson. *Applied Integer Programming: Modeling and Solution*. Wiley, 2010.
6. G. Di Battista, W. Didimo, M. Patrignani, and M. Pizzonia. Orthogonal and quasi-upward drawings with vertices of prescribed size. In J. Kratochvíl, editor, *Graph Drawing*, volume 1731 of *LNCS*, pages 297–310. Springer, 1999.
7. G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 1998.
8. C. A. Duncan and M. T. Goodrich. Graph drawing and cartography. In R. Tamassia, editor, *Handbook of Graph Drawing and Visualization*, chapter 7, pages 223–246. CRC Press, 2013.
9. M. Eiglsperger. *Automatic Layout of UML Class Diagrams: A Topology-Shape-Metrics Approach*. PhD thesis, Universität Tübingen, 2003.
10. M. Eiglsperger, S. P. Fekete, and G. W. Klau. Orthogonal graph drawing. In M. Kaufmann and D. Wagner, editors, *Drawing Graphs, Methods and Models*, volume 2025 of *Lecture Notes in Computer Science*, pages 121–171. Springer, 1999.
11. M. Eiglsperger, U. Fößmeier, and M. Kaufmann. Orthogonal graph drawing with constraints. In D. B. Shmoys, editor, *SODA*, pages 3–11. ACM/SIAM, 2000.
12. M. Eiglsperger and M. Kaufmann. Fast Compaction for Orthogonal Drawings with Vertices of Prescribed Size. In P. Mutzel, M. Jünger, and S. Leipert, editors, *Graph Drawing*, volume 2265 of *Lecture Notes in Computer Science*, pages 124–138. Springer Berlin Heidelberg, 2002.
13. U. Fößmeier and M. Kaufmann. Drawing high degree graphs with low bend numbers. In F. Brandenburg, editor, *Graph Drawing*, volume 1027 of *LNCS*, pages 254–266. Springer, 1995.
14. Gurobi Optimization, Inc. Gurobi Optimizer Reference Manual, 2014. `http://www.gurobi.com`.
15. C. E. Leiserson. Area-efficient graph layouts. In *FOCS*, pages 270–281. IEEE, 1980.
16. R. Tamassia. On embedding a graph in the grid with the minimum number of bends. *SIAM Journal of Computing*, 16(3):421–444, 1987.
17. L. G. Valiant. Universality considerations in VLSI circuits. *IEEE Transactions on Computers*, 100(2):135–140, 1981.

Appendix: Additional Experimental Evaluation Results

In this section, we present the results of our experimental evaluation for the test sets of the planar Rome graphs and the planar North graphs. They clearly show that no or not much progress can be expected without many triangular faces (only 12% for the Rome graphs and only 13% for the North graphs on average).

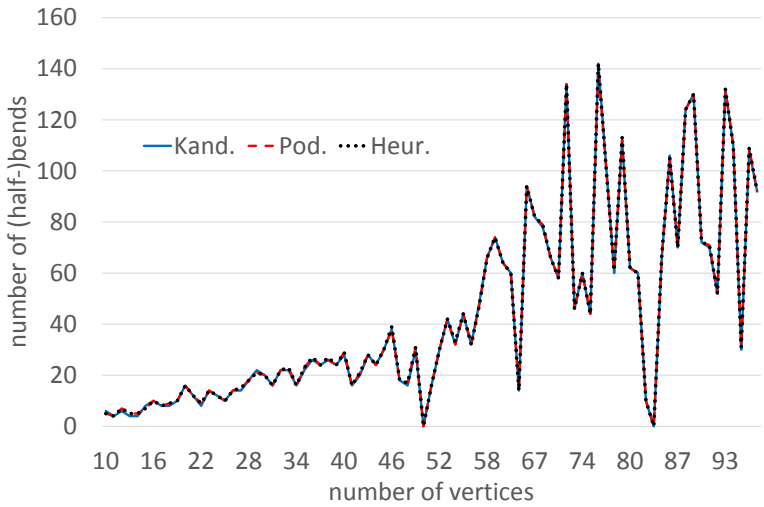


(a) Number of vertices against total number of bends.

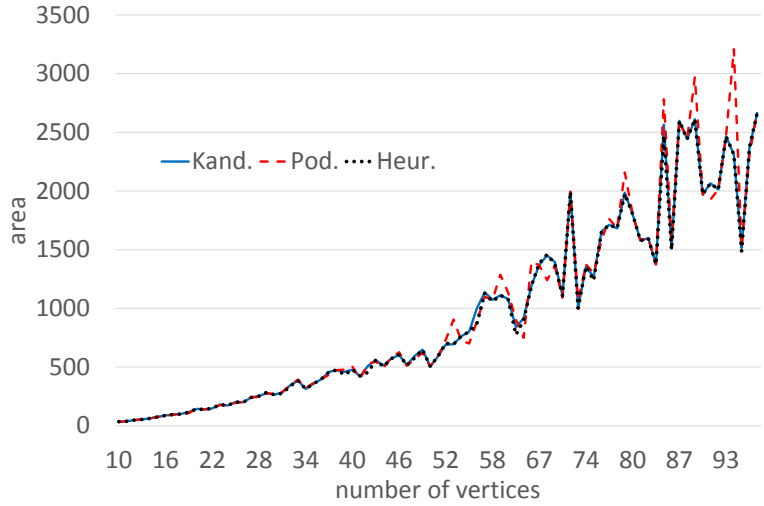


(b) Number of vertices against area requirements.

Fig. 8. The results of our experimental evaluation for the test set of the planar Rome graphs.



(a) Number of vertices against bend number.



(b) Number of vertices against area.

Fig. 9. The results of our experimental evaluation for the test set of the planar North graphs.

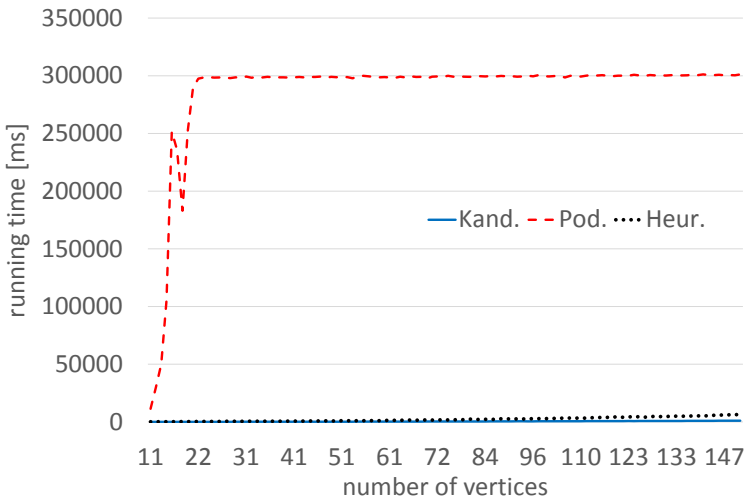


Fig. 10. Running time results for the test set of the randomly created triangulations.