Anchored Drawings of Planar Graphs[⋆]

Patrizio Angelini[1], Giordano Da Lozzo[1], Marco Di Bartolomeo[1],
Giuseppe Di Battista[1], Seok-Hee Hong[2], Maurizio Patrignani[1], and Vincenzo Roselli[1]

[1] Department of Engineering, Roma Tre University, Italy
{angelini,dalozzo,dibartolomeo,gdb,
patrigna,roselli}@dia.uniroma3.it
[2] School of Information Technologies, The University of Sydney, Australia
shhong@it.usyd.edu.au

Abstract. In this paper we study the ANCHORED GRAPH DRAWING (AGD) problem: Given a planar graph G, an initial placement for its vertices, and a distance d, produce a planar straight-line drawing of G such that each vertex is at distance at most d from its original position.

We show that the AGD problem is NP-hard in several settings and provide a polynomial-time algorithm when d is the uniform distance L_∞ and edges are required to be drawn as horizontal or vertical segments.

1 Introduction

Several applications require to draw graphs whose vertices are constrained to be not too much *distant* from specific points [1,9]. As an example, consider a graph whose vertices are cities and whose edges are relationships between cities. It is conceivable that the user wants to draw the graph on a geographic map where vertices have the coordinates of the corresponding cities. Unfortunately, depending on the local density of the cities, the drawing may be cluttered or may contain crossings between edges that might disappear if the vertices could move from their locations. Hence, the user may be interested to trade precision for quality of the drawing, accepting that the vertices move of a certain distance from the location of the cities, provided that the readability of the drawing increases. Problems in which the input consists of a set of imprecise points have also been studied in Computational Geometry [4,7].

In this paper we consider the following problem, that we call ANCHORED GRAPH DRAWING (AGD)[1]. Given a graph $G = (V, E)$, an initial placement for its vertices, and a distance δ, we ask whether there exists a planar drawing of G, according to a certain drawing convention, such that each vertex $v \in V$ can move at distance at most δ from its initial placement. Note that the problem can have different formulations depending on how the concepts of "readability" and "distance" are defined.

We consider both straight-line planar drawings and rectilinear planar drawings. Further, in addition to the traditional L_2 Euclidean distance, we consider the L_1 Manhattan

[⋆] Work partially supported by ESF EuroGIGA GraDR, by the MIUR project AMANDA "Algorithmics for MAssive and Networked DAta", prot. 2012C4E3KT_001, and by the EU FP7 STREP Project "Leone: From Global Measurements to Local Management", no. 317647.

[1] We remark that the term 'anchored graph' was used within a different setting in [3].

Table 1. The complexity of the ANCHORED GRAPH DRAWING problem depending on the metric and drawing style adopted when the areas of the vertices do not overlap

Metric	**Distance**	**Region Shape**	**Straight-line**	**Rectilinear**
L_1	Manhattan	$\diamond$	NP-hard	NP-hard
L_2	Euclidean	$\bigcirc$	NP-hard	NP-hard
L_∞	Uniform	$\square$	NP-hard	Polynomial

distance and the L_∞ 'uniform' distance. Note that, adopting L_2 distance is equivalent to allowing vertices to be placed into circular regions centered at their original positions, and adopting L_1 or L_∞ distances is equivalent to allowing vertices to be placed into diamond-shaped or square-shaped areas, respectively.

Observe that, if the regions of two vertices overlap, the positions of the two vertices can be swapped with respect to their initial placement, which may be confusing to a user of the drawing. Moreover, overlapping between vertex regions would make problem AGD as difficult as known Clustered Planarity variants, such as the Strip Planarity problem [2] in the straight-line setting, whose complexity is a non-trivial open problem. Hence, we restrict to instances such that the regions of the vertices do not overlap.

We remark that the version of the problem where each circle may have a different size was shown to be NP-hard in [6] by reducing Planar-$(3,4)$-SAT with variable repetitions (where repeated occurrences of one variable in one clause are counted repeatedly). The proof in [6] uses disks with radius zero and disks with large radii. Also, the reduction relies on overlapping disks.

Furthermore, we observe that the NP-hardness of the problem with different distances and overlapping areas trivially follows from the NP-hardness of extending a planar straight-line drawing [10] by setting $\delta(v) = 0$ for each fixed vertex v and allowing suitably large distances for vertices that have to be planarly added to the drawing.

In this paper we show that the ANCHORED GRAPH DRAWING problem is NP-hard for any combination of metrics and drawing standards that we considered, with the exception of rectilinear drawings and uniform distance metric (square-shaped regions). These results, summarized in Table 1, were somehow unexpected, as computing a planar rectilinear drawing of a graph, without any further constraint, is NP-hard [5].

The paper is organized as follows. Section 2 contains basic definitions and terminology. Section 3 describes a polynomial-time algorithm when the considered distance is the uniform distance L_∞ and edges are required to be drawn as either horizontal or vertical segments. Section 4 is devoted to the NP-hardness proofs of all the other considered settings of the problems. Finally, Section 5 discusses some open problems.

2 Problem Definition and Instances Classification

A *straight-line planar drawing* of a graph G is a drawing of G where edges are straight-line segments that do not intersect except at common end-points. A *rectilinear planar drawing* is a straight-line planar drawing where edges are parallel to the axes.

Given two points p and q in the plane, denote by $dx(p,q)$ and $dy(p,q)$ the differences of their coordinates, i.e., $dx(p,q) = |x(p) - x(q)|$ and $dy(p,q) = |y(p) - y(q)|$, where

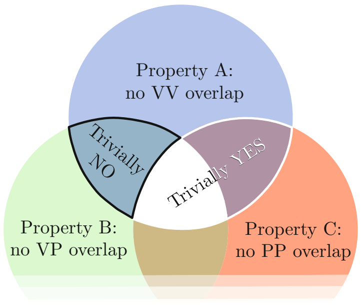

Fig. 1. Venn diagram describing the logical relationships among Properties A–C

$x(r)$ and $y(r)$ are the x- and y-coordinate of a point r, respectively. The *Euclidean distance* $d_2(p,q)$ of p and q is defined as $d_2(p,q) = (dx(p,q)^2 + dy(p,q)^2)^{\frac{1}{2}}$. The *Manhattan distance* is defined as $d_1(p,q) = dx(p,q) + dy(p,q)$. The *uniform distance* $d_\infty(p,q) = \lim_{i\to\infty}(dx(p,q)^i + dy(p,q)^i)^{\frac{1}{i}} = \max(dx(p,q), dy(p,q))$.

We define the ANCHORED GRAPH DRAWING problem parametrically in the metric L_k and the drawing style $\mathcal{X}$, which can be straight-line ($\mathcal{X} = \mathcal{S}$) or rectilinear ($\mathcal{X} = \mathcal{R}$). Hence, for any $L_k \in \{L_1, L_2, L_\infty\}$ and any $\mathcal{X} \in \{\mathcal{S}, \mathcal{R}\}$ we define: **Problem:** ANCHORED GRAPH DRAWING-L_k-$\mathcal{X}$ (AGD-L_k-$\mathcal{X}$). **Instance:** A graph $G = (V,E)$, an initial placement for its vertices $\alpha(v) : V \to \Re^2$, and a distance δ. **Question:** Does there exist a planar drawing of G according to the $\mathcal{X}$ drawing convention such that each vertex $v \in V$ is at distance L_k at most δ from $\alpha(v)$?

We define *anchored drawing* a planar drawing satisfying all the requirements of the particular version of problem ANCHORED GRAPH DRAWING.

Given an instance $\langle G, \alpha, \delta \rangle$ of the ANCHORED GRAPH DRAWING problem, each vertex v identifies a region $R(v)$ of the plane, called *vertex region*, that encloses the initial position of the vertex and whose shape depends on the metric adopted for computing the distance. In particular, for the Euclidean distance the vertex regions are circles, for the Manhattan distance they are diamonds, and for the uniform distance they are squares. Each edge (u,v) of the graph, instead, identifies a *pipe* $P(u,v)$, defined as follows. Consider the convex hull H of $R(u)$ and $R(v)$; pipe $P(u,v)$ is the closed region obtained by removing $R(u)$ and $R(v)$ from H.

Instances can be classified based on the intersections among vertex and pipe regions. Namely, we can have instances satisfying the following properties:
Property A. No overlap between two vertex regions (VV-overlaps);
Property B. No overlap between a vertex region and a pipe (VP-overlaps);
Property C. No overlap between pipes (PP-overlaps) not incident to the same vertex.

The Venn diagram in Fig. 1 shows the logical relationships between the three properties. The following observation is immediate.

Observation 1. *If Properties A, B, and C are all satisfied, then the instance is trivially positive, since choosing any point in the vertex region (including the initial placement of the vertex) yields an anchored drawing of the input graph.*

In this paper we always assume that Property A is satisfied. In fact, if vertex regions were allowed to overlap, then it would be possible to reduce to this problem a variant of the Clustered Planarity problem whose complexity is still unknown. In this variant, which includes Strip Planarity [2] as a special case, the cluster regions are already drawn and edges are straight-line.

Two further observations can be made which reduce the set of instances of interest.

Observation 2. *An instance satisfying Property B but not satisfying Property C (i.e., with PP-overlaps but without VP-overlaps) is trivially false, as in this case any PP-overlap would enforce a crossing between two edges for any placement of their end-vertices in the corresponding vertex regions.*

Observation 3. *An instance satisfying Property C but not satisfying Property B (i.e., with VP-overlaps but without PP-overlaps) is trivially true.*

Proof: Since Property C holds, no crossing can occur outside a vertex region. First, suppose that regions are diamonds or squares. If the center of region $R(v)$ of a vertex v lies inside a pipe $P(x, y)$, then at least two consecutive vertices, say a and b, delimiting $R(v)$ lie inside $P(x, y)$. This implies that v has degree at most 1, as otherwise there would be a PP-overlap between $P(x, y)$ and a pipe $P(v, w)$ delimited by either a or b.

As for the case in which regions are circles, if the center of $R(v)$ lies inside $P(x, y)$, then at least half of the circle delimiting $R(v)$ lies inside $P(x, y)$. Hence, a similar argument applies to prove that $\deg(v) \leq 1$.

In all the three cases, since $\deg(v) \leq 1$ and $R(v)$ is not completely contained into $P(x, y)$, v can be placed on any point of $R(v)$ outside $P(x, y)$. Hence, placing each other vertex at the center of its region yields an anchored drawing. □

Due to the above properties and observations, the remaining part of this paper focuses on the instances for which Property A holds, while Properties B and C do not. These instances correspond to the blue region at the top of Fig. 1.

3 Polynomial-Time Algorithm

In this section we describe an algorithm, called **Algo-AGD-L_∞-$\mathcal{R}$**, that decides in polynomial time instances $\langle G, \alpha, \delta \rangle$ of problem AGD-L_∞-$\mathcal{R}$ such that G is connected.

For each vertex $v \in V$, denote by $x_l(v)$ and $x_r(v)$ the x-coordinate of the left and right side of $R(v)$, respectively. Similarly, denote by $y_t(v)$ and $y_b(v)$ the y-coordinate of the top and bottom side of $R(v)$, respectively. See region $R(u)$ in Fig. 2.

First note that, for each edge $(u, v) \in E$, the relative placement of $R(u)$ and $R(v)$ determines whether (u, v) has to be drawn as a vertical or a horizontal segment, or (u, v) cannot be drawn neither horizontal nor vertical with its endpoints lying inside their corresponding regions. In the latter case, instance I is negative. An edge that has to be drawn as a horizontal (vertical) segment is a *horizontal* (*vertical*) edge. In the following we assume w.l.o.g. that any horizontal edge (u, v) is such that $x_r(u) < x_l(v)$, while any vertical edge (u, v) is such that $y_t(u) < y_b(v)$. A path composed only of horizontal (vertical) edges is a *horizontal* (*vertical*) path. Given that each edge (u, v)

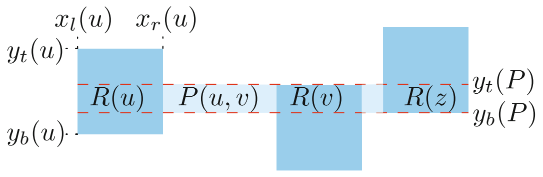

Fig. 2. Geometric description of a region $R(u)$ and of a pipe $P(u,v)$, after procedure PIPEEQUALIZER has been applied

can be categorized as either horizontal or vertical, we can label its pipe $P(u,v)$ as either `horizontal` or `vertical` accordingly. Also, we can determine the minimum and maximum y-coordinate (x-coordinate) that a horizontal (vertical) edge (u,v) can assume while placing both its endvertices inside their regions. In the following we describe pipe $P(u,v)$ by means of these coordinates, which are denoted by $y_b(P)$ and $y_t(P)$ ($x_l(P)$ and $x_r(P)$), respectively. See `horizontal` pipe $P(u,v)$ in Fig. 2.

Also note that, if a vertex v of degree 2 is incident to two horizontal (vertical) edges (u,v) and (v,z), then replacing v and its incident edges with a horizontal (vertical) edge (u,z) yields an equivalent instance. Hence, we assume that, if there exists a vertex of degree 2, then it is incident to both a horizontal and a vertical edge.

As a preliminary step of the algorithm, we initialize the geometric description of each pipe $P(u,v)$ as follows. If P is `vertical`, then set $x_r(P) = min(x_r(u), x_r(v))$ and $x_l(P) = max(x_l(u), x_l(v))$. If P is `horizontal`, then set $y_t(P) = min(y_t(u), y_t(v))$ and $y_b(P) = max(y_b(u), y_b(v))$. Here and in the following, whenever a vertex region $R(w)$ (a pipe $P(u,v)$) is modified by the algorithm, we assume the pipes incident to w (the regions $R(u)$ and $R(v)$) to be modified accordingly.

In order to ensure that `horizontal` (`vertical`) pipes whose edges belong to the same horizontal (vertical) path have the same geometric description, we refine the pipes by applying the following procedure, that we call PIPEEQUALIZER. As long as there exist two `vertical` pipes $P'(u,v)$ and $P''(v,w)$ incident to the same vertex v such that $x_l(P') \neq x_l(P'')$ or $x_r(P') \neq x_r(P'')$, set $x_l(P') = x_l(P'') = max(x_l(P'), x_l(P''))$ and $x_r(P') = x_r(P'') = min(x_r(P'), x_r(P''))$. Analogously, as long as there exist two `horizontal` pipes $P'(u,v)$ and $P''(v,w)$ incident to the same vertex v such that $y_b(P') \neq y_b(P'')$ or $y_t(P') \neq y_t(P'')$, set $y_b(P') = y_b(P'') = max(y_b(P'), y_b(P''))$ and $y_t(P') = y_t(P'') = min(y_t(P'), y_t(P''))$. See pipe $P(u,v)$ in Fig. 2 after the application of PIPEEQUALIZER.

We then perform the following procedure, that we call PIPECHECKER. It first checks whether there exists a pipe P such that $x_r(P) < x_l(P)$ or $y_t(P) < y_b(P)$. Then, it checks whether there exists a PP-overlap between two pipes $P(u,v)$ and $P(w,z)$ such that: (i) neither of $R(u)$ or $R(v)$ has a VP-overlap with $P(w,z)$; and (ii) neither of $R(w)$ or $R(z)$ has a VP-overlap with $P(u,v)$. If one of the two checks succeeds, then we conclude that instance I is negative, otherwise we proceed with the algorithm.

In the following, every time a pipe is modified, we will apply procedure PIPEEQUALIZER to extend this modification to other pipes, and procedure PIPECHECKER to test whether such modifications resulted in uncovering a negative instance.

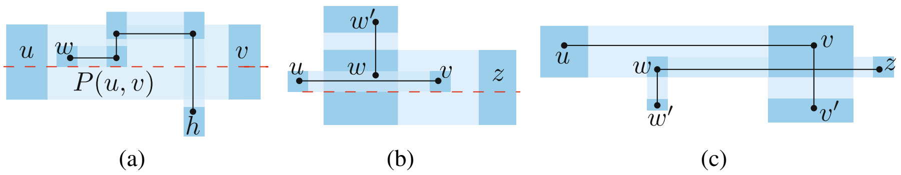

Fig. 3. Vertices exiting pipes. (a) Vertex w exits $P(u, v)$ from below. The cut of $P(u, v)$ applied by procedure PIPEBLOCKCHECKER is described by a dashed line. (b) Vertex v exits $P(w, z)$ through w. The cut of $R(w)$ and the consequent cut of $P(w, z)$ applied by procedure VERTEXCHECKER is described by a dashed line. (c) A situation recognized by procedure PIPEINTERLEAVECHECKER.

The general strategy of the main part of the algorithm is to progressively reduce the size of the pipes. In particular, at each step we consider the current instance I^i and modify it to obtain an instance I^{i+1} with smaller pipes than I^i that admits an anchored drawing if and only if I^i admits an anchored drawing. Eventually, such a process will lead either to an instance I^m for which it is easy to construct an anchored drawing or to conclude that instance $I = I^1$ is negative.

Let $P(u, v)$ be a `horizontal` pipe, and w be a vertex having a VP-overlap with $P(u, v)$. Refer to Fig. 3(a). We say that w *exits* P *from below* if there exists a vertex h such that: (i) $y_b(P) < y_b(w) < y_t(P)$ and $x_r(u) < x_l(w) < x_r(w) < x_l(v)$; (ii) $y_t(h) < y_b(P)$ and $x_r(u) < x_l(h) < x_r(h) < x_l(v)$; and (iii) there exists a path $\gamma = (w, \ldots, h)$ in G connecting w to h in which every internal vertex r is such that $R(r)$ intersects P. Symmetrically, we say that w *exits* P *from above* if there exists a vertex h with the same properties as before, except for the fact that $y_b(P) < y_t(w) < y_t(P)$, $y_b(h) > y_t(P)$. Otherwise, we say that w *exits* P *through a vertex*, either u or v. In Fig. 3(b), vertex v exits pipe $P(w, z)$ through w. Observe that, since G is connected and no VV-overlap occurs in I, there always exists a path $\gamma = (w, \ldots, h)$ in G connecting w to a vertex h such that h does not have any VP-overlap with P; hence, w always exits P, either from above or below, or through a vertex.

For the case of a `vertical` pipe $P(u, v)$, we assume analogous definitions of vertices exiting P *from left, right* or *through a vertex*, either u or v. As long as one of the following conditions is satisfied, we apply one of the procedures described hereunder.

***Procedure* VERTEXCHECKER:** Consider a vertex w having a VP-overlap with a `horizontal` (`vertical`) pipe $P(u, v)$ such that $y_b(w) \leq y_b(P) < y_t(P) \leq y_t(w)$ (resp., $x_l(w) \leq x_l(P) < x_r(P) \leq x_r(w)$). If w is incident to two `vertical` (`horizontal`) pipes, then we conclude that instance I is negative. Otherwise, if w is incident to a `vertical` (`horizontal`) pipe $P(w, w')$, then set $y_b(w) = max(y_b(w), y_b(P))$ (set $x_l(w) = max(x_l(w), x_l(P))$. See Fig. 3(b). Analogously, if w is incident to a `vertical` (`horizontal`) pipe $P(w', w)$, then set $y_t(w) = min(y_t(w), y_t(P)$ (set $x_r(w) = min(x_r(w), x_r(P))$.

***Procedure* PIPEBLOCKCHECKER:** Consider a pipe $P(u, v)$ having a VP-overlap with a vertex w such that w does not exit through a vertex. If w exits $P(u, v)$ both from above and from below (a `vertical` pipe both from left and from right), then we conclude that instance I is negative. Otherwise, if w exits P from (i) above, we set $y_t(P) = y_t(w)$; (ii) below, we set $y_b(P) = y_b(w)$; (iii) left, we set $x_l(P) = x_l(w)$; or (iv) right, we set $x_r(P) = x_r(w)$. See Fig. 3(a).

***Procedure* PIPESIDECHECKER:** Consider a `horizontal` (`vertical`) pipe $P(u, v)$ and a vertex w exiting $P(u, v)$ both through vertex u and through vertex v. If u and v are incident to `vertical` (`horizontal`) pipes, either $P(u, u')$ and $P(v', v)$, or $P(u', u)$ and $P(v, v')$, respectively, then we conclude that instance I is negative.

***Procedure* PIPEINTERLEAVECHECKER:** Suppose that there exist two `horizontal` (`vertical`) pipes $P(u, v)$ and $P(w, z)$ such that v and $P(w, z)$ have a VP-overlap, and w and $P(u, v)$ have a VP-overlap. If either v is incident to a `vertical` (`horizontal`) pipe $P(v, v')$ and w is incident to a `vertical` (`horizontal`) pipe $P(w, w')$, or v is incident to a `vertical` (`horizontal`) pipe $P(v', v)$ and w is incident to a `vertical` (`horizontal`) pipe $P(w', w)$, then we conclude that instance I is negative. See Fig. 3(c).

If none of the above procedures can be applied, then we conclude that I is a positive instance.

Theorem 1. *Let $I = \langle G, \alpha, \delta \rangle$ be an instance of* AGD-L_∞-$\mathcal{R}$ *such that G is connected. Algorithm* **Algo-AGD-L_∞-$\mathcal{R}$** *decides in polynomial time whether $\langle G, \alpha, \delta \rangle$ admits an anchored drawing.*

Proof: The initialization of the pipes and their refinement operated by procedure PIPEEQUALIZER, both after the initialization and after each further modification, is trivially necessary to meet the requirements that vertices are placed inside their regions and edges are drawn as either horizontal or vertical segments.

Suppose that procedure PIPECHECKER concluded that instance I is negative at some point of the algorithm. If $x_r(P) < x_l(P)$ (if $y_t(P) < y_b(P)$), then there exist two `vertical` (`horizontal`) pipes sharing a vertex that cannot be placed inside its region while drawing both its incident edges as rectilinear segments. Otherwise, there exists a PP-overlap between two pipes $P(u, v)$ and $P(w, z)$ not overlapping with regions $R(u)$, $R(v)$, $R(w)$, and $R(z)$. By Observation 2, the instance is negative.

We prove that the modifications operated by VERTEXCHECKER, when a vertex w has a VP-overlap with a `horizontal` (`vertical`) pipe $P(u, v)$ and w is incident to a `vertical` (`horizontal`) $P(w, w')$, do not restrict the possibility of constructing an anchored drawing of $\langle G, \alpha, \delta \rangle$. Refer to Fig. 3(b). In fact, in this case, in any anchored drawing of $\langle G, \alpha, \delta \rangle$, edge (w, w') cannot traverse $P(u, v)$ from top to bottom. As for the fact that an instance in which w is incident to two `vertical` (`horizontal`) pipes is correctly recognized as negative, observe that in this case one of the two vertical edges incident to w necessarily crosses edge (u, v).

We prove that the modifications operated by PIPEBLOCKCHECKER, when a vertex w overlaps a pipe $P(u, v)$ and does not exit through one of its vertices, do not restrict the possibility of constructing an anchored drawing of $\langle G, \alpha, \delta \rangle$. Suppose that w exits $P(u, v)$ from below, the other cases being analogous. Refer to Fig. 3(a). The statement

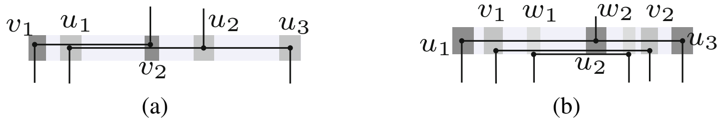

Fig. 4. Construction of the drawing when none of the procedures can be applied. (a) Two maximal horizontal paths (u_1, u_2, u_3) and (v_1, v_2) whose pipes have the same y-coordinates. Path (v_1, v_2) is assigned a y-coordinate slightly larger than (u_1, u_2, u_3). (b) Three maximal horizontal paths (u_1, u_2, u_3), (v_1, v_2), and (w_1, w_2)whose pipes have the same y-coordinates. Path (v_1, v_2) is assigned a y-coordinate slightly larger than (w_1, w_2).

follows from the fact that, in any anchored drawing of $\langle G, \alpha, \delta\rangle$, the drawing of path $\gamma = (w, \dots, h)$ blocks visibility from $R(u)$ to $R(v)$ inside $P(u, v)$ at least for all the y-coordinates in the range between the point where w is placed and the point where h is placed. Since $y_t(h) < y_b(P)$, the point where w is placed determines a new lower bound for the value of $y_b(P)$. Since such a point cannot be below $y_b(w)$, the statement follows. As for the fact that an instance containing a vertex w that exits a `horizontal` pipe both from above and from below (a `vertical` pipe both from left and from right) is correctly recognized as negative, observe that in this case the two paths starting from w completely block visibility between from $R(u)$ to $R(v)$ inside $P(u, v)$.

We prove that an instance containing a vertex w that exits $P(u, v)$ through both of its vertices, and such that u and v are also incident to pipes $P(u, u')$ and $P(v', v)$, or vice versa, reaching them from different sides, is correctly recognized as negative by procedure PIPESIDECHECKER. Namely, observe that in this case path (u', u, v, v') necessarily crosses one of the two paths starting from w.

Finally, suppose that procedure PIPEINTERLEAVECHECKER concluded that instance I is negative. Refer to Fig. 3(c). It is easy to observe that the fact that v and w are reached from the same side is not compatible with an anchored drawing of $\langle G, \alpha, \delta\rangle$.

We conclude the proof of the theorem by showing that, when none of the described procedures can be applied, it is always possible to draw every edge (u, v) inside its pipe $P(u, v)$, as follows.

Consider every maximal horizontal path $(u_1, \dots, u_r)$. Note that, each vertex u_i, with $1 \leq i \leq r$, is incident to at least a `vertical` pipe, either (u_i, u_i') or (u_i', u_i), as otherwise edges (u_{i-1}, u_i) and (u_i, u_{i+1}) would have been replaced with edge (u_{i-1}, u_{i+1}). If all the vertices u_i are incident to a `vertical` (u_i, u_i'), then assign y-coordinate equal to $y_t(u_i)$ to u_i, for $i = 1, \dots, r$; if all the vertices u_i are incident to a `vertical` (u_i', u_i), then assign y-coordinate equal to $y_b(u_i)$ to u_i, for $i = 1, \dots, r$; finally, if there exists at least a vertex u_i incident to a `vertical` (u_i, u_i') and at least a vertex u_j incident to a `vertical` (u_j', u_j), then assign y-coordinate equal to $\frac{y_b(u_i)+y_t(u_i)}{2}$ to u_i, for $i = 1, \dots, r$. Assign x-coordinates to vertices of every maximal vertical path equal to $x_l(u_i)$, to $x_r(u_i)$, or to $\frac{x_l(u_i)+x_r(u_i)}{2}$, in an analogous way.

With a straightforward case analysis, it is possible to observe that, since none of the conditions activating the described procedures is satisfied, there exists no crossing in the drawing, apart from possible overlaps between edges belonging to different maximal horizontal (vertical) paths whose pipes have the same bottom and top y-coordinates (the same left and right x-coordinates). However, these overlaps can be always eliminated by increasing (decreasing) of an arbitrarily small amount the coordinates of the overlapping paths (see two examples in Fig. 4(a) and 4(b)), again due to the fact that none of the conditions activating the described procedures is satisfied. □

4 Hardness Results

In this section we prove the hardness of the ANCHORED GRAPH DRAWING problem in different settings. In particular, Theorem 2 is devoted to the hardness of the AGD-L_2-$\mathcal{S}$ problem, i.e., the problem of generating planar straight-line drawings of the input graph where the vertex regions are circles of radius δ. Theorem 3, instead, is devoted to the hardness of the AGD-L_2-$\mathcal{R}$ problem, where the regions are circles of radius δ and edges are required to be drawn as horizontal or vertical segments.

The proofs of hardness for the remaining variants of the problem listed in Table 1 can be derived from these two and, thus, will not be explained in detail. Namely, the reduction to AGD-L_1-$\mathcal{R}$ is very similar to that used for AGD-L_2-$\mathcal{R}$, and can be obtained by suitably replacing circles with diamond-shaped regions that ensure analogous geometric visibility and obstruction properties. The same holds for the hardness proof of AGD-L_∞-$\mathcal{S}$, that can be obtained from AGD-L_2-$\mathcal{S}$ with small adaptations of the gadgets. Finally, the reduction to AGD-L_1-$\mathcal{S}$ is the same as the one for AGD-L_∞-$\mathcal{S}$ where all the geometric constructions are rotated by 45^o, transforming the square-shaped regions of AGD-L_∞-$\mathcal{S}$ into the diamond-shaped regions of AGD-L_1-$\mathcal{S}$.

All our proofs are based on a reduction from the NP-complete problem PLANAR 3-SATISFIABILITY [8], defined as follows. **Problem:** PLANAR 3-SATISFIABILITY (P3SAT). **Instance:** A planar bipartite graph $G = (V_v, V_c, E)$ where: (i) V_v is a set of variables; (ii) V_c is a set of clauses, each consisting of exactly three literals representing variables in V_v; and (iii) E is a set of edges connecting each variable $v \in V_v$ to all the clauses containing a literal representing v. **Question:** Does there exist a truth assignment to the variables so that each clause has at least one `true` literal?

For each of our problems, we describe gadgets that, given an instance ϕ of P3SAT, can be combined to construct an instance γ of the considered problem. Namely, we describe a gadget for each of the following: *variable*, *not*, *turn*, *split*, and *clause*.

The variable gadget has two families of planar drawings, corresponding to the two truth values. The not gadget admits planar drawings that invert its input truth value. The turn gadget admits planar drawings that propagate its input truth value in a direction that is orthogonal to the original one. The split gadget admits planar drawings that propagate its input truth value to two different directions. Finally, the clause gadget is planar if and only if at least one of its input literals is `true`. The gadgets are combined following the structure of a planar drawing of ϕ, so that any planar drawing of γ corresponds to a truth assignment for the variables satisfying ϕ. Similarly, given a truth assignment for the variables that satisfies ϕ, the gadgets for variables can be drawn accordingly to obtain a planar drawing of γ.

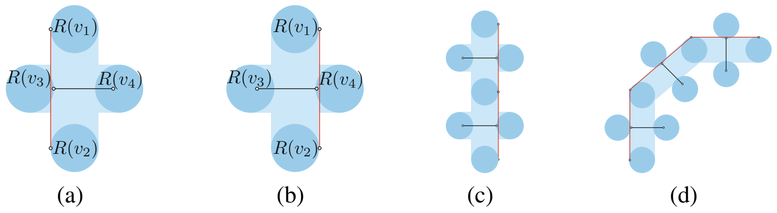

Fig. 5. *Variable* gadget for the reduction to the AGD-L_2-$\mathcal{S}$ problem in its false (a) and true (b) configurations. (c) Propagation of the true configuration of a variable gadget. (d) *Turn* gadget in its false configuration.

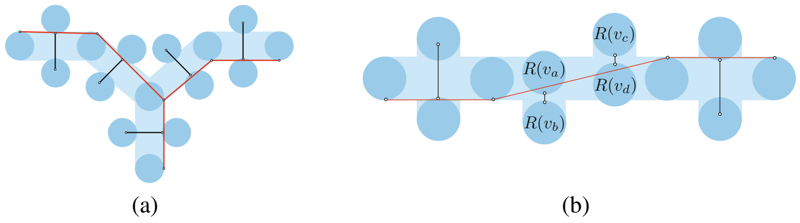

Fig. 6. (a) *Split* gadget in its true configuration. (b) *Not* gadget.

Theorem 2. AGD-L_2-$\mathcal{S}$ *is NP-hard.*

Proof: To prove hardness we reduce problem P3SAT to AGD-L_2-$\mathcal{S}$, under the hypothesis that Property A is satisfied (no overlap among vertex regions).

Let ϕ be an instance of P3SAT with n variables and m clauses. We describe how to construct an equivalent instance γ of AGD-L_2-$\mathcal{S}$. For each variable x_i, $i = 1, \dots, n$, we create a *variable* gadget, whose two families of planar drawings are depicted in Figs. 5(a) and 5(b), consisting of four vertices v_1, v_2, v_3, and v_4 and edges (v_1, v_2) and (v_3, v_4). The regions assigned to the vertices are placed as follows: (i) the centers of $R(v_1)$ and of $R(v_2)$ lie on the same vertical line; (ii) the centers of $R(v_3)$ and of $R(v_4)$ lie on the same horizontal line; (iii) pipe $P(v_1, v_2)$ has an intersection of arbitrarily small area with both $R(v_3)$ and $R(v_4)$; and (iv) pipe $P(v_3, v_4)$ intersects neither $R(v_1)$ nor $R(v_2)$. Hence, in any anchored drawing of γ, edge (v_1, v_2) is drawn either to the left of v_3 (as in Fig. 5(a)) or to the right of v_4 (as in Fig. 5(b)). In both cases, edge (v_1, v_2) is drawn almost vertical. We call these two configurations *false* and *true* configurations for the variable gadget, respectively, and associate them with the `false` and `true` values for the corresponding variable x_i. The truth value of a variable can be propagated by concatenating a sequence of variable gadgets $\mu_1, \dots, \mu_k$ in which $R(v_1)$ of μ_i is identified with $R(v_2)$ of μ_{i+1}, for each $i = 1, \dots, k-1$. See Fig. 5(c).

The *turn* gadget can be constructed by concatenating three variable gadgets, μ_1, μ_2 and μ_3, as depicted in Fig. 5(d), in such a way that μ_2 has a clockwise rotation of $45°$ with respect to μ_1, and μ_3 has a clockwise rotation of $90°$ with respect to μ_1.

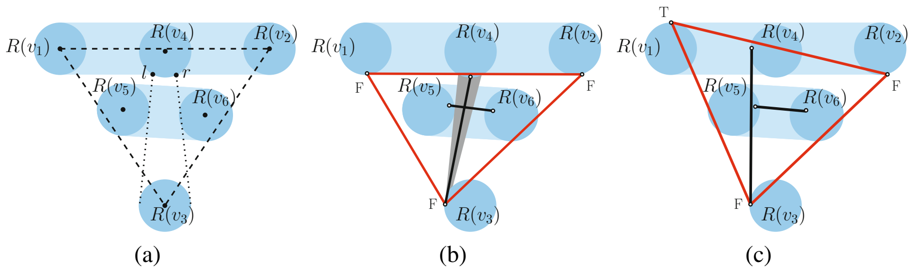

Fig. 7. Clause for the reduction to the AGD-L_2-$\mathcal{S}$ problem. Vertices v_1, v_2, and v_3 represent the three literals of the clause. For readability, we show only pipes $P(v_1, v_2)$ and $P(v_5, v_6)$. (a) Arrangement of the regions. (b) All literals are assigned `false`, and edges (v_5, v_6) and (v_3, v_4) cross. The darker wedge represents all the possible positions for edge (v_3, v_4) in this truth assignment, which implies that the crossing is unavoidable. (c) Assigning `true` to any of the literals allows for a planar drawing.

The *split* gadget can be constructed by combining two turn gadgets $\tau_L = \langle\mu_1^L, \mu_2^L, \mu_3^L\rangle$ and $\tau_R = \langle\mu_1^R, \mu_2^R, \mu_3^R\rangle$, with $\mu_1^L = \mu_1^R$ and where τ_L is obtained from τ_R by a vertical mirroring. See Fig. 6(a).

The *not* gadget is constructed as follows. Consider two horizontally (vertically) aligned variable gadgets μ_1 and μ_2. Add an edge connecting v_2 of μ_1 to v_1 of μ_2, as in Fig. 6(b). Place between μ_1 and μ_2 two pairs of adjacent vertices (v_a, v_b) and (v_c, v_d) whose regions are placed in such a way that: (i) any drawing of edge (v_a, v_b) blocks the visibility between the true configurations of μ_1 and μ_2; (ii) any drawing of edge (v_c, v_d) blocks the visibility between the false configurations of μ_1 and μ_2; and (iii) edges (v_a, v_b) and (v_c, v_d) can be drawn in such a way that there exists visibility between different truth configurations of μ_1 and μ_2. Hence, in any anchored drawing of γ, the configurations of μ_1 and μ_2 are different.

The *clause* gadget is constructed as follows. Refer to Fig. 7(a). Consider three vertices v_1, v_2, and v_3 whose regions are placed in such a way that their centers induce a non-degenerated triangle $\mathcal{T}$ and the centers of $R(v_1)$ and of $R(v_2)$ lie on the same horizontal line. These three vertices represent the three literals of the clause. While $R(v_2)$ and $R(v_3)$ maintain the usual convention to encode the truth value of the represented variable, for $R(v_1)$ it is inverted. It can be easily realized by negating the value of the variable. The gadget contains three more vertices: v_4, v_5, and v_6, and edges (v_1, v_2), (v_2, v_3), (v_1, v_3), (v_3, v_4), and (v_5, v_6). The center of $R(v_i)$, with $i = 4, 5, 6$, lies inside $\mathcal{T}$. Region $R(v_4)$ is completely contained in pipe $P(v_1, v_2)$, except for an arbitrarily small part Π, which lies inside $\mathcal{T}$. Consider the two points l and r in which the boundary of $R(v_4)$ intersects $P(v_1, v_2)$. The boundary of $R(v_5)$ is tangent to the leftmost segment of the convex hull H of $\{l, r\} \cup R(v_3)$. Region $R(v_6)$ completely lies to the right of the rightmost segment of H, except for an arbitrarily small part. Neither $R(v_5)$ nor $R(v_6)$ intersects $P(v_1, v_2)$.

If all the literals are set to `false`, then v_4 must lie below edge (v_1, v_2) (and hence in Π). However, visibility between Π and $R(v_3)$ is prevented by edge (v_5, v_6) (Fig. 7(b)).

Otherwise, if at least one of the literals is set to `true`, then an anchored drawing of γ can be realized (see Fig. 7(c) for an example). □

Theorem 3. AGD-L_2-$\mathcal{R}$ *is NP-hard.*

Proof sketch: To prove hardness we reduce problem P3SAT to AGD-L_2-$\mathcal{R}$, under the hypothesis that Property A is satisfied (no overlap among vertex regions). The adopted gadgets are similar to those used in the proof of Theorem 2 with the exception of the clause gadget. That gadget is based on creating three horizontal strips that are the only possible containers of a specific edge. If all the literals are `false`, then suitable edges obstruct such strips and make it not possible to construct an anchored drawing. □

5 Conclusions and Open Problems

We considered the Anchored Graph Drawing problem in several settings, showing that, provided that the input instance do not have overlaps between vertex regions (Property A), the problem of producing planar drawings is NP-hard in most of the settings. The only exception is for the case with rectilinear drawings and uniform distances (square-shaped regions), for which a polynomial-time algorithm is provided in Section 3.

We leave open the following questions: (i) Does problem AGD belong to class NP? (ii) The instances in our NP-hardness proofs can be augmented to equivalent instances whose graphs are biconnected (we omit details for space reasons). In these instances, different truth values correspond to different embeddings. What is the complexity of AGD when the input graph is triconnected or has at least a fixed embedding? (iii) What if we allow the vertex regions to (at least partially) overlap?

References

1. Abellanas, M., Aiello, A., Hernández, G., Silveira, R.I.: Network drawing with geographical constraints on vertices. In: Actas XI Encuentros de Geom. Comput., pp. 111–118 (2005)
2. Angelini, P., Da Lozzo, G., Di Battista, G., Frati, F.: Strip planarity testing. In: Wismath, S., Wolff, A. (eds.) GD 2013. LNCS, vol. 8242, pp. 37–48. Springer, Heidelberg (2013)
3. Cabello, S., Mohar, B.: Adding one edge to planar graphs makes crossing number and 1-planarity hard. SIAM J. Comput. 42(5), 1803–1829 (2013)
4. Dumitrescu, A., Mitchell, J.S.B.: Approximation algorithms for TSP with neighborhoods in the plane. J. Algorithms 48(1), 135–159 (2003)
5. Garg, A., Tamassia, R.: On the computational complexity of upward and rectilinear planarity testing. SIAM J. Comput. 31(2), 601–625 (2001)
6. Godau, M.: On the difficulty of embedding planar graphs with inaccuracies. In: Tamassia, R., Tollis, I.G. (eds.) GD 1994. LNCS, vol. 894, pp. 254–261. Springer, Heidelberg (1995)
7. Löffler, M., van Kreveld, M.J.: Largest and smallest convex hulls for imprecise points. Algorithmica 56(2), 235–269 (2010)
8. Lichtenstein, D.: Planar formulae and their uses. SIAM J. Comput. 11, 185–225 (1982)
9. Lyons, K.A., Meijer, H., Rappaport, D.: Algorithms for cluster busting in anchored graph drawing. J. Graph Algorithms Appl. 2(1) (1998)
10. Patrignani, M.: On extending a partial straight-line drawing. International Journal of Foundations of Computer Science (IJFCS) 17(5), 1061–1069 (2006)