Simply Realising an Imprecise Polyline is NP-hard

Simply Realising an Imprecise Polyline is NP-hard

Thijs van der Horst[1,2], Tim Ophelders[1,2], and Bart van der Steenhoven[2]

1 **Department of Information and Computing Sciences, Utrecht University, the Netherlands**
{t.w.j.vanderhorst|t.a.e.ophelders}@uu.nl

2 **Department of Mathematics and Computer Science, TU Eindhoven, the Netherlands**
b.j.v.d.steenhoven@student.tue.nl

Abstract

We consider the problem of deciding, given a sequence of regions, if there is a choice of points, one for each region, such that the induced polyline is simple or weakly simple, meaning that it can touch but not cross itself. Specifically, we consider the case where each region is a translate of the same shape. We show that the problem is NP-hard when the shape is a unit-disk or unit-square. We argue that the problem is is NP-complete when the shape is a vertical unit-segment.

1 Introduction

Finding a planar drawing of a graph has been studied widely due to its many applications, for example in floor-planning and visualisations [7]. Often when the graphs come from geometric data, such as road networks, vertices already have predetermined locations. In these cases, if edges must be drawn as straight line segments, there is a unique corresponding drawing. Real-world data such as GPS trajectories are often imprecise. Imprecision can be modeled by assigning to each vertex an imprecision region. A geometric graph induced by assigning each vertex to a point in its imprecision region is called a *realisation*.

Algorithms for imprecise data have been considered in the past [5, 8]. We consider the problem of deciding whether imprecise data admits a realisation without self-intersections. Godau [4] showed that this problem is NP-hard if the imprecision regions are closed disks of varying radius. In follow-up work, Angelini et al. [2] showed that the problem remains NP-hard when the regions are pairwise-disjoint unit-disks, or unit-squares. In this work, we consider the problem for a very restricted graph class, namely path-graphs. That is, we are looking for a realisation that is a simple polyline.

Efficient algorithms exist to determine whether a polygon or polyline is weakly simple [1, 3]. When the polygon is imprecise however, Löffler [6] showed that determining whether a weakly simple realisation exists when the imprecision regions are scaled translates of a fixed shape, such as a segment or circle, is NP-hard. Finally, Silveira et al. [9] show that the problem is NP-complete for straight-line drawings of graphs where each vertex has degree exactly one (i.e. the graph is a matching), using unit-length vertical segments as imprecision regions.

Definitions and problem statement. A *polygonal chain* or *polyline* is a piecewise-linear path in the plane, given by a sequence $P = (p_i)_{i=0}^n$ of points called vertices. With slight abuse of notation, we interpret P as the polyline, instead of its vertices. An *imprecise polyline* is given by a sequence of regions $R = (r_i)_{i=0}^n$ in the plane. We call the polyline $(p_i)_{i=0}^n$ a *realisation* of R if $p_i \in r_i$ for each i.

39th European Workshop on Computational Geometry, Barcelona, Spain, March 29–31, 2023.
This is an extended abstract of a presentation given at EuroCG'23. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

A polyline is *simple* (or plane) if it does not cross or touch itself. An *ε-perturbation* of a polyline $P = (p_i)_{i=0}^n$ is a polyline $P' = (p'_i)_{i=0}^n$ such that $\|p_i - p'_i\| \leq \varepsilon$ for all i. A polyline $(p_i)_{i=0}^n$ is *weakly simple* if for every $\varepsilon > 0$ there is an ε-perturbation that is simple. We are interested in the problem of deciding whether an imprecise polyline R admits a weakly simple realisation.

Results and organisation. We show that the problem is NP-hard already when each region is a translate of the same shape S. This setting differs from [6] in the sense that now all regions must have uniform size. In Section 2 we prove this for the case where S is a unit disk, and discuss how the construction can be adapted to show NP-hardness for unit squares. Hence, computing the infimum ε for which a polyline has a simple ε-perturbation under the L_1, L_2, and L_∞ norms is NP-hard. In the full version we additionally show NP-completeness if S is a unit-length vertical segment.

2 Unit disks as regions

We prove by reduction from planar monotone 3SAT that the problem is NP-hard when all regions are unit disks. We first introduce planar monotone 3SAT. A monotone 3CNF formula is a formula $\varphi(x_1, \ldots, x_m) := \bigwedge_i C_i$ with Boolean variables $x_1, \ldots, x_m$, where each clause C_i is either positive (of the form $(x_j \vee x_k \vee x_l)$) or negative (of the form $(\neg x_j \vee \neg x_k \vee \neg x_l)$). A monotone rectilinear layout of a monotone 3CNF formula is a plane drawing of the incidence graph between its clauses and variables, in which variables are drawn as disjoint horizontal segments on the x-axis, all positive clauses lie above the x-axis, all negative clauses lie below the x-axis, and edges are rectilinear, do not cross the x-axis, and have at most one bend. We may assume for each clause that its central edge does not bend, see Figure 1. Planar monotone 3SAT is an NP-complete problem that asks, given a monotone 3CNF formula φ with a given monotone rectilinear layout, whether φ is satisfiable. For our reduction we construct gadgets consisting of imprecise polylines corresponding to variables, clauses, and edges of the layout. We then show how to connect these gadgets into a single imprecise polyline that has a weakly simple realisation if and only if the formula φ if satisfiable.

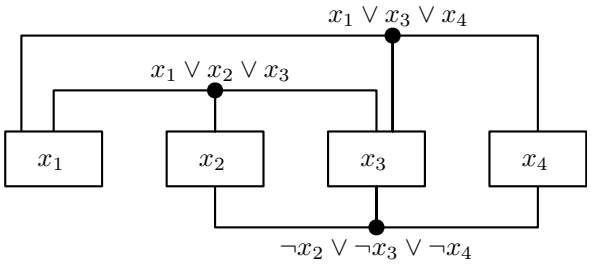

Figure 1 Example of an induced variable-clause graph for planar monotone 3SAT.

Pivot gadget. Before we construct the gadgets for the variables and clauses, we introduce an auxiliary gadget: the *pivot gadget*. The purpose of the pivot gadget is to force a given edge of any weakly simple realisation to go through some fixed point of our choosing. The gadget is illustrated in Figure 2 and consists of three components. Two of these components together form the pivot so that the edge of the third component must pass through the

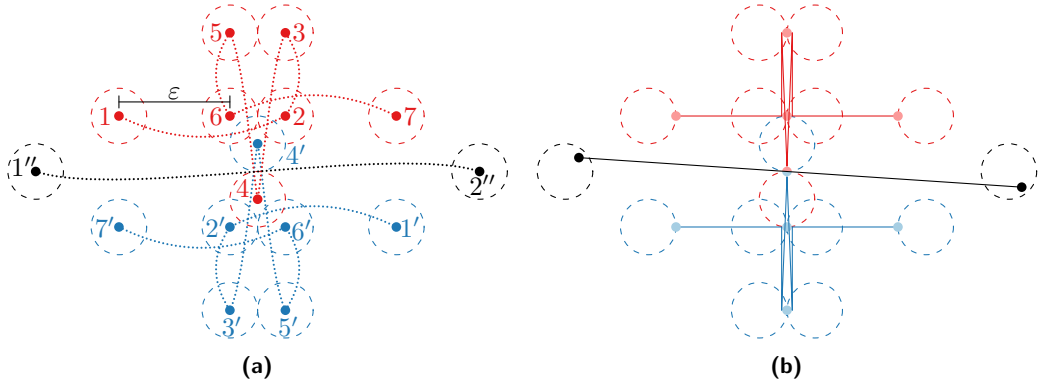

Figure 2 (a) The pivot gadget. (b) A weakly simple realisation, showing a simple perturbation. Note that this perturbation moves most vertices outside of their assigned disks

center $(0, 0)$ of the pivot. The coordinates of the top component will be:

$$1, \ldots, 7 \mapsto (-1-\varepsilon, 2), (1, 2), (1, 5), (0, -1), (-1, 5), (-1, 2), (1+\varepsilon, 2)$$

for some value $\varepsilon > 0$, and the bottom component is a rotated copy of the top component:

$$1', \ldots, 7' \mapsto (1+\varepsilon, -2), (-1, -2), (-1, -5), (0, 1), (1, -5), (1, -2), (-1-\varepsilon, -2).$$

There is some freedom in choosing the placement of regions $1''$ and $2''$, but to make sure that the realisation passes through the pivot we require that the x-coordinate of region $1''$ is less than -1, that of region $2''$ is greater than 1, and the tangent lines to the pair of circles $1''$ and $2''$ all intersect the segment between $(0, -5)$ and $(0, 5)$. We can accommodate any angle of the edge between regions $1''$ and $2''$ by choosing $\varepsilon > 0$ sufficiently small. Namely, such that the point $(-\varepsilon, 2)$ where we will realise region 1 lies above the tangent lines from point $(0, 0)$ to region $1''$, and such that symmetric properties hold for regions 7, $1'$ and $7'$.

Figure 2(b) depicts a weakly simple realisation of the pivot gadget. Other weakly simple realisations may differ in placement for points in regions 1, 7, $1'$, $7'$, $1''$ and $2''$ only; the segment connecting $1''$ and $2''$ must always pass through $(0, 0)$.

▶ **Lemma 1.** *For any weakly simple realisation, the segment of the pivot gadget connecting regions* $1''$ *and* $2''$ *must pass through* $(0, 0)$.

▶ **Lemma 2.** *Any choice of points for regions* $1''$ *and* $2''$ *such that the segment connecting them passes through* $(0, 0)$ *has a corresponding weakly simple realisation of the pivot gadget.*

Variable gadget. We construct a variable gadget that can take on two distinct states, one for true and one for false, see Figure 3(a). The exact coordinates of the regions are as follows,

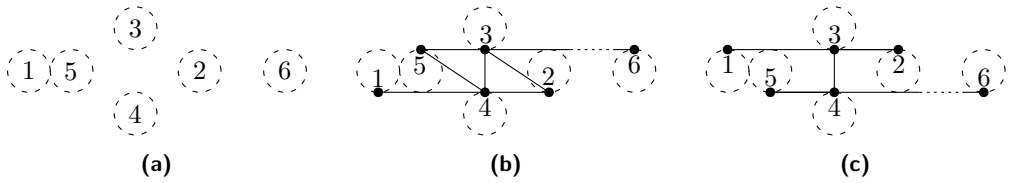

Figure 3 (a) The variable gadget. (b) The false state. (c) The true state.

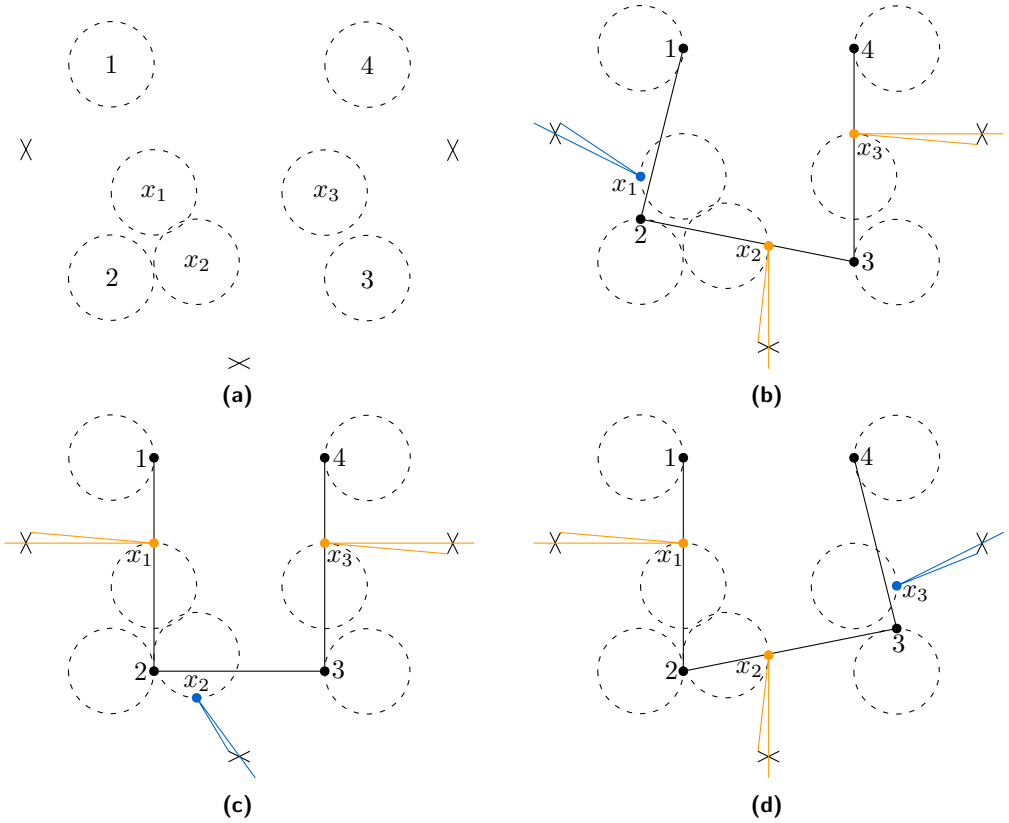

Figure 4 (a) The clause gadget, with realisations where (b) x_1, (c) x_2, or (d) x_3 is true.

assuming the left most region is centered at $(0, 0)$ and $l > 8$ is arbitrary:

$$1, \ldots, 6 \mapsto (0, 0), (8, 0), (5, 2), (5, -2), (2, 0), (l, 0).$$

▶ **Lemma 3.** *The variable gadget admits exactly two weakly simple realisations.*

If the final horizontal line from region 5 to 6 is above the blocking vertical line between regions 3 and 4, then the gadget is in its false state (Figure 3(b)). We consider the situation where it is below as the true state (Figure 3(c)).

Clause gadget. The clause gadget is used to represent formulas of the form $(x_1 \vee x_2 \vee x_3)$, where the variables can also all be negated. The initial placement of the regions for this gadget can be seen in Figure 4(a). We use an X-shape to schematically represent the pivot gadgets. The regions x_1, x_2 and x_3 correspond to the literals of the clause and the regions 1, 2, 3 and 4 are used to put restrictions on these literals. The exact positions of the regions of this gadget are as follows, centering the top-left region at $(0, 0)$:

$$1, 2, 3, 4 \mapsto (0, 0), (0, -5), (6, -5), (6, 0).$$

We place the point for x_1 at $(1, -3)$, x_2 at $(2, -4.6)$ and x_3 at $(5, -3)$. The construction used to connect the regions of these literals to variable gadgets and the exact placement of the pivot gadgets is explained later. For now, it is only important to know that when a variable is false, then the part of the polyline going to the literal point must either go

completely vertical through the pivot gadget at the bottom, or completely horizontal through the pivot gadgets at the sides. The regions of x_1, x_2, and x_3 are placed precisely so that in this situation there is exactly one realisation that does not intersect the pivot gadget. We shall refer to the position of a variable in this realisation as its *false position* and say that it is *covered* in a solution if it lies in the interior of the polyline defined by the corner points of the clause gadget. If the variable is true then the line can be placed at an angle through the pivot gadget, allowing the literal point to be realised in multiple ways. We refer to the left-, bottom- and rightmost positions of x_1, x_2, and x_3 as their respective *true position*.

▶ **Lemma 4.** *For every choice of two of the three literals there exists a realisation that uncovers their false positions, and any realisation uncovers at most two false positions.*

Connecting variables and clauses. For the reduction we must connect the variable gadgets to the clause gadgets. For this we follow the layout given by the planar monotone 3SAT instance (see Figure 1). Clauses always connect to variables in a downward or upward facing E-shape. To preserve the planarity of this graph, our reduction should follow this shape to a close enough degree. So the clause gadget is to be placed directly above one of the variable gadgets and the other two variable gadgets connect to it with a bend from the side.

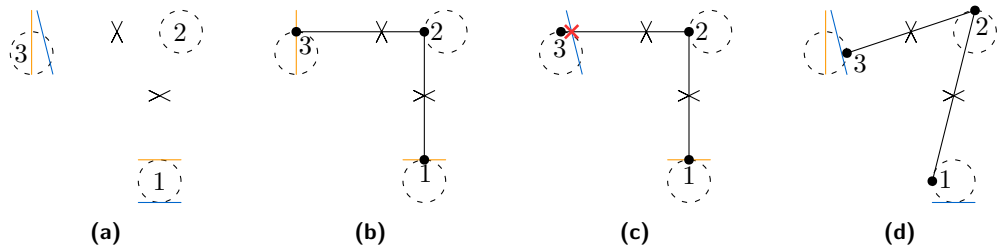

Figure 5 (a) The right wire gadget to connect a variable to a clause. Realisations where (b) both are in the false state, (c) the variable is in its false state and the clause in its true state and (d) the variable gadget in its true state.

Consider the right connection from a variable to a clause. The wire gadget in this case involves three regions and two pivot gadgets (see Figure 5(a)). Region 1 is horizontally aligned with the variable gadget of the corresponding to the right literal of the clause. As in Figure 3, we place these variable regions between region 2 and 6, resulting in a realisation with a horizontal line at the top if the state is false and at the bottom if the state is true. We denote the position of region 1 as $(0, 0)$. Region 3 is the right literal region of the clause gadget. The position of this region is determined by the clause gadget. Let region 3 be positioned at $(-a, b)$. For our wire gadget, we place region 2 at position $(1, b + 1)$ and place two pivot gadgets at $(\frac{-a}{2} + 1, b + 1)$ and $(0, \frac{b}{2} + 1)$, oriented appropriately. We stretch the variable-clause graph without changing its validity, so that a and b are large enough that the components of the wire gadget do not overlap.

▶ **Lemma 5.** *If a variable gadget for x_i is in its false state, any weakly simple realisation places x_i in its false position. Otherwise x_i can be placed in either its true or false position.*

The wire gadget for connecting the left literal of a clause to its variable gadget is the above gadget but mirrored. For connecting the middle variable to the clause we do the same, except here we need only a single pivot gadget and no region 2, as this connection does not need to bend. All three versions of the wire gadget can be seen in effect in Figure 6.

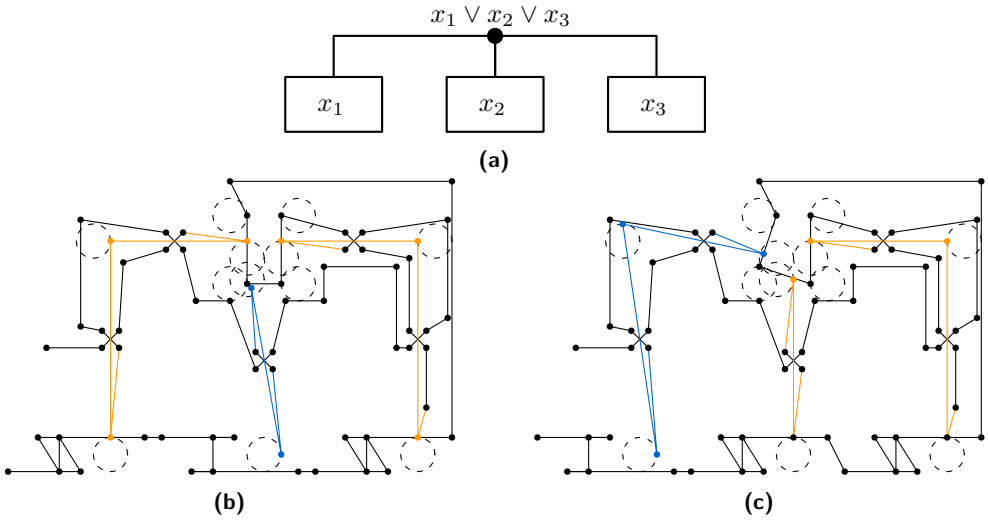

Figure 6 Example of the construction for variable-clause graph (a). The weakly simple realisation in (b) sets x_2 to true and the realisation in (c) sets x_1 to true.

The complete reduction. The general construction works as follows. First we place all gadgets as prescribed by the layout obtained from the planar monotone 3SAT instance. Next, we connect the gadgets in a planar manner. For this, we connect the variables from left to right as in Figure 6. We connect the gadgets in the top half of the construction by processing the clauses on the outer face (that lie above the x-axis) as follows, and connecting the results from left to right. Connect the clause to its right wire (and its pivot gadgets), then recursively process and connect the clauses in the region enclosed by its right and middle wire, then connect the middle wire, then recursively process and connect the clauses in the region enclosed by the middle and left wire, and finally connect the left wire. The gadgets in the bottom half are connected in a symmetric manner. We can put everything together by connecting the variables, top half, and bottom half (their loose endpoints all lie on the "outer face").

Correctness of the reduction follows from the correctness of the individual gadgets. As each gadget and connection uses a constant number of regions, whose positions can be determined based on the variable-clause graph, the reduction is polynomial in size and time.

▶ **Theorem 6.** *Deciding whether there exists a weakly simple realisation for a sequence of unit disks is NP-hard.*

▶ **Observation 7.** If our construction admits a weakly simple realisation, then there is one that uses only the top, right, bottom and leftmost points of its regions. The problem hence remains NP-hard when each region is a subset of a unit disk and contains these four points.

The problem we considered can also be looked at from a different point of view. One could consider the center of each region in the sequence to be a vertex of an input polyline and the regions themselves denoting how we are allowed to perturb these vertices. In that case the setting of unit disks that we discussed corresponds to perturbing vertices by unit distance under the L_2 norm. Under the L_1 norm the regions are diamond-shaped and contained in a unit disk. Thus by the above observation the problem remains NP-hard. After an affine transformation, it follows that the problem is also NP-hard under the L_∞ norm.

3 Discussion

We showed that deciding if an imprecise polygon has a weakly simple realisation is NP-hard if each region is a unit disk, unit square, or vertical unit segment. By growing each region slightly, the same follows for deciding whether a simple realisation exists. Whereas the case of vertical segments is NP-complete, it is unclear whether the problems of Section 2 lie in NP, as it is unclear whether polynomially many bits suffice to encode a solution. The settings of Section 2 lie in the complexity class $\exists\mathbb{R}$, and we wonder if these settings are $\exists\mathbb{R}$-hard.

References

1 Hugo A. Akitaya, Greg Aloupis, Jeff Erickson, and Csaba D. Tóth. Recognizing weakly simple polygons. *Discret. Comput. Geom.*, 58(4):785–821, 2017.

2 Patrizio Angelini, Giordano Da Lozzo, Marco Di Bartolomeo, Giuseppe Di Battista, Seok-Hee Hong, Maurizio Patrignani, and Vincenzo Roselli. Anchored drawings of planar graphs. In *GD*, volume 8871 of *Lecture Notes in Computer Science*, pages 404–415. Springer, 2014.

3 Hsien-Chih Chang, Jeff Erickson, and Chao Xu. Detecting weakly simple polygons. In *SODA*, pages 1655–1670. SIAM, 2015.

4 Michael Godau. On the difficulty of embedding planar graphs with inaccuracies. In *GD*, volume 894 of *Lecture Notes in Computer Science*, pages 254–261. Springer, 1994.

5 Maarten Löffler. *Data Imprecision in Computational Geometry*. PhD thesis, Utrecht University, 2009.

6 Maarten Löffler. Existence and computation of tours through imprecise points. *Int. J. Comput. Geom. Appl.*, 21(1):1–24, 2011.

7 Takao Nishizeki and Md. Saidur Rahman. *Planar Graph Drawing*, volume 12 of *Lecture Notes Series on Computing*. World Scientific, 2004.

8 Marcel Roeloffzen. Finding structures on imprecise points. Master's thesis, TU Eindhoven, 2009.

9 Rodrigo I. Silveira, Bettina Speckmann, and Kevin Verbeek. Non-crossing paths with geographic constraints. *Discret. Math. Theor. Comput. Sci.*, 21(3), 2019.