On d-regular Schematization of Embedded Paths

Andreas Gemsa[1], Martin Nöllenburg[⋆1,2], Thomas Pajor[1], and Ignaz Rutter[1]

[1] Institute of Theoretical Informatics, Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany
[2] Department of Computer Science, University of California, Irvine, USA

Abstract. In the d-regular path schematization problem we are given an embedded path P (e.g., a route in a road network) and an integer d. The goal is to find a *d-schematized* embedding of P in which the orthogonal order of all vertices in the input is preserved and in which every edge has a slope that is an integer multiple of $90°/d$. We show that deciding whether a path can be d-schematized is NP-hard for any integer d. We further model the problem as a mixed-integer linear program. An experimental evaluation indicates that this approach generates reasonable route sketches for real-world data.

1 Introduction

Angular or $\mathscr{C}$-oriented schematizations of graphs refer to a class of graph drawings, in which the admissible edge directions are limited to a given set $\mathscr{C}$ of (usually evenly spaced) slopes. This includes the well-known class of orthogonal drawings and extends more generally to k-linear drawings, e.g., octilinear metro maps. Applications of schematic drawings can be found in various domains such as cartography, VLSI layout, and information visualization.

In many schematization scenarios the input is not just a graph but a graph with an initial drawing that has to be schematized according to the given set of slopes. This is the case, e.g., in cartography, where the geographic positions of network vertices and edges are given [6], in sketch-based graph drawing, where a sketch of a drawing is given and the task is to improve or schematize that sketch [3], or in dynamic graph drawing, where each drawing in a sequence of drawings must be similar to its predecessor [5]. For such a redrawing task it is crucial that the mental map [13] of the user is preserved, i.e., the output drawing must be as similar as possible to the input. Misue et al. [13] suggested preserving the *orthogonal order* of the input drawing as a simple criterion for maintaining a set of basic spatial properties of the input, namely the relative above/below and left/right positions of all pairs of input nodes. The orthogonal order has been used successfully as a means for maintaining the mental map [4, 7, 9, 11].

The motivation behind the work presented here is the visualization of routes in road networks as sketches for driving directions. An important property of a route sketch is that it focuses on road changes and important landmarks rather than exact geography and distances. Typically the start and destination lie in populated areas that are locally reached via a sequence of relatively short road segments. On the other hand, the majority

⋆ Supported by grant NO 899/1-1 of the German Research Foundation (DFG)

of the route typically consists of long highway segments with no or only few road changes. This property makes it difficult to display driving directions for the whole route in a single traditional map since some areas require much smaller scales than others. The strength of route sketches for this purpose is that they are not drawn to scale but rather use space proportionally to the route complexity.

Related Work. Geometrically, we can consider a route to be an embedded path in the plane. The simplification of paths (or polylines) in cartography is well studied and the classic line simplification algorithm by Douglas and Peucker [8] is one of the most popular methods. Two more recent algorithms were proposed for $\mathscr{C}$-oriented line simplification [12, 14]. These line simplification algorithms, however, are not well suited for drawing route sketches since they keep the positions of the input points fixed or within small local regions around the input points and thus edge lengths are more or less fixed. On the other hand, Agrawala and Stolte [1] presented a system called *LineDrive* that uses heuristic methods based on simulated annealing to draw route sketches. Their method allows distortion of edge lengths and angles. It does not, however, restrict the set of edge directions and does not give hard quality guarantees for the mental map such as the preservation of the orthogonal order.

A graph drawing problem that has similar constraints as drawing route sketches is the metro-map layout problem, in which an embedded graph is to be redrawn octilinearly. The problem is known to be NP-hard [15] but it can be solved successfully in practice by mixed-integer linear programming [16]. The existing methods covered in a survey by Wolff [17] do aim to keep the mental map of the input but no strict criterion like the orthogonal order is applied. Brandes and Pampel [4] studied the path schematization problem in the presence of orthogonal order constraints in order to preserve the mental map. They showed that deciding whether a rectilinear schematization exists that preserves the orthogonal order of the input path is NP-hard. They also showed that schematizing a path using arbitrarily oriented unit-length edges is NP-hard. Delling et al. [7] gave an efficient algorithm to compute $\mathscr{C}$-oriented drawings of monotone paths that preserve the orthogonal order and have minimum schematization cost. The schematization cost counts the number of edges that are not drawn with their closest $\mathscr{C}$-oriented direction. The authors also sketch a heuristic approach for schematizing non-monotone paths.

Contributions. In this paper we close the complexity gap of the path schematization problem that remained open between the hardness result of Brandes and Pampel [4] for rectilinear paths and the efficient algorithm of Delling et al. [7] for monotone $\mathscr{C}$-oriented paths. We prove that deciding whether a $\mathscr{C}$-oriented orthogonal-order preserving drawing of an embedded input path exists is NP-hard, even if the path is simple. This is true for every *d-regular* set $\mathscr{C}$ of slopes that have angles that are integer multiples of $90°/d$ for any integer d. The case $d = 1$ is covered by Brandes and Pampel [4] but their proof relies on the absence of diagonal edges and hence does not extend to other values of d. We show the hardness in the octilinear case $d = 2$ in Section 3 and subsequently, in Section 4, how this result extends to the general d-regular case for $d > 2$. We finally design and evaluate a mixed integer-linear program (MIP) for solving the d-regular path schematization problem in Section 5. Our experimental results show that routes in practice usually consist of only a small number of relevant road segments and that our MIP is indeed able to quickly generate reasonable sketches for those routes. Omitted proofs and details about the MIP are found in the appendix.

2 Preliminaries

A plane embedding of a graph $G = (V,E)$ is a mapping $\pi : G \to \mathbb{R}^2$ that maps every vertex $v \in V$ to a distinct point $\pi(v) = (x_\pi(v), y_\pi(v))$ and every edge $e = uv \in E$ to the line segment $\pi(e) = \overline{\pi(u)\pi(v)}$ such that no two edges e_1, e_2 cross in π except at common endpoints. For simplicity we also use the terms vertex and edge to refer to their images under an embedding.

We measure the slope of an edge uv as the counterclockwise angle formed between the horizontal line through $\pi(u)$ and the line segment $\pi(uv)$. For a set of angles $\mathscr{C}$ we say that a drawing is *$\mathscr{C}$-oriented* if the slope of every edge $e \in E$ is contained in $\mathscr{C}$. A set of slopes $\mathscr{C}$ is called *d-regular* for an integer d if $\mathscr{C} = \mathscr{C}_d = \{i \cdot 90°/d \mid i \in \mathbb{Z}\}$.

Let π and ρ be two embeddings of the same graph G. We say that ρ respects the *orthogonal order* [4] of π if for any two vertices u and $v \in V$ it holds that $x_\rho(u) \leq x_\rho(v)$ if $x_\pi(u) \leq x_\pi(v)$ and $y_\rho(u) \leq y_\rho(v)$ if $y_\pi(u) \leq y_\pi(v)$. In other words, the orthogonal order defines the relative above-below and left-right positions of any two vertices.

Let (G,π) be a graph G with a plane input embedding π. A *d-regular schematization* (or *d-schematization*) of (G,π) is a plane embedding ρ that is $\mathscr{C}_d$-oriented, that preserves the orthogonal order of π and where no two vertices are embedded at the same coordinates. We also call ρ *valid* if it is a d-schematization of (G,π).

3 Hardness of 2-regular Path Schematization

In this section we show that the problem of deciding whether there is a 2-schematization for a given embedded graph (G,π) is NP-hard, even if G is a simple path. In the latter case we denote the problem as the (d-regular) PATH SCHEMATIZATION PROBLEM (PSP). We fix $d = 2$. To prove that 2-regular PSP is NP-hard we first show hardness of the closely related 2-regular UNION OF PATHS SCHEMATIZATION PROBLEM (UPSP), where (G,π) is a set $\mathscr{P}$ of k embedded disjoint paths $\mathscr{P} = \{P_1,\ldots,P_k\}$.

We show that 2-regular UPSP is NP-hard by a reduction from MONOTONE PLANAR 3-SAT, which is known to be NP-hard [2]. MONOTONE PLANAR 3-SAT is a special variant of PLANAR 3-SAT where each clause either contains exactly three positive literals or exactly three negative literals and additionally, the variable-clause graph admits a planar drawing such that all variables are on the x-axis, the positive clauses are embedded below the x-axis and the negative clauses above the x-axis. An example instance of MONOTONE PLANAR 3-SAT is depicted in Fig. 1. In a second step, we show how to augment the set of paths $\mathscr{P}$ to form a single simple path P that has the same properties as $\mathscr{P}$ and thus proves that PSP is NP-hard.

In the following, we assume that φ is a given MONOTONE PLANAR 3-SAT instance with variables $X = \{x_1,\ldots,x_n\}$ and clauses $C = \{c_1,\ldots,c_m\}$.

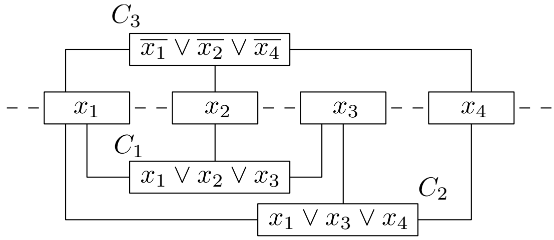


Fig. 1. A MONOTONE PLANAR 3-SAT instance with four variables and three clauses.

3.1 Hardness of the UNION OF PATHS SCHEMATIZATION PROBLEM

In the following we first introduce the different types of required gadgets and then show how to combine them in order to prove the hardness of UPSP.

Border Gadget. For all our gadgets we need to control the placement of vertices on discrete positions. Thus our first gadget is a path whose embedding is unique up to scaling and translation. This path will induce a grid on which we subsequently place the remaining gadgets.

Let $\Delta_x(u,v)$ be the x-distance between the two vertices u and v. Likewise, let $\Delta_y(u,v)$ be the y-distance between u and v. We call a simple path P with embedding π *rigid* if a 2-schematization of (P,π) is unique up to scaling and translations. Further, we call an embedding π of P *regular* if there exists a length $\ell > 0$ such that for any two vertices u, v of P it holds that $\Delta_x(u,v) = z_x \cdot \ell$ and $\Delta_y(u,v) = z_y \cdot \ell$ for some $z_x, z_y \in \mathbb{Z}$. Thus in a regular embedding of P all vertices are embedded on a grid whose cells have side length ℓ. For our border gadget we construct a simple path B of appropriate length with embedding π that is both rigid and regular. Hence, π is essentially the unique 2-schematization of (B,π), and after rescaling we can assume that all points of B lie on an integer grid. The border gadget consists of a horizontal component B_h and a vertical component B_v which share a common starting vertex $v_1 = v'_1$, see Fig. 2. The component B_h alternates between a 45° edge to the upper right and a vertical edge downwards. The vertices are placed such that their y-coordinates alternate and hence all odd, respectively all even, vertices have the same y-coordinate. The vertical component consists of a copy of the horizontal component rotated by 90° in clockwise direction around v_1. To form B, we connect B_v and B_h by identifying their starting points v_1 and v'_1.

Lemma 1. *The border gadget B with its given embedding π is rigid and regular.*

In the following we define the length of a grid cell induced by B as 1 so that we obtain an integer grid. We choose B long enough to guarantee that any vertex of our subsequent gadgets lies vertically and horizontally between two pairs of vertices in B.

The grid in combination with the orthogonal order gives us the following properties:

1. If we place a vertex v with two integer coordinates, i.e., on a grid point, its position in any valid embedding is unique;
2. if we place a vertex v with one integer and one non-integer coordinate, i.e., on a grid edge, its position in any valid embedding is on that grid edge;

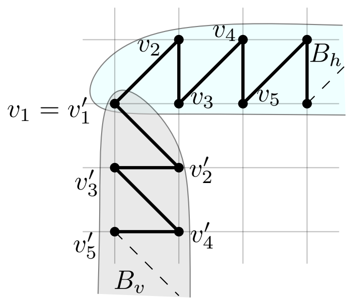


Fig. 2. Border gadget B that is rigid and regular. The horizontal component is denoted by B_h and the vertical one by B_v.

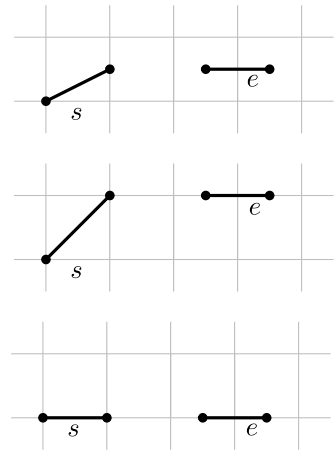


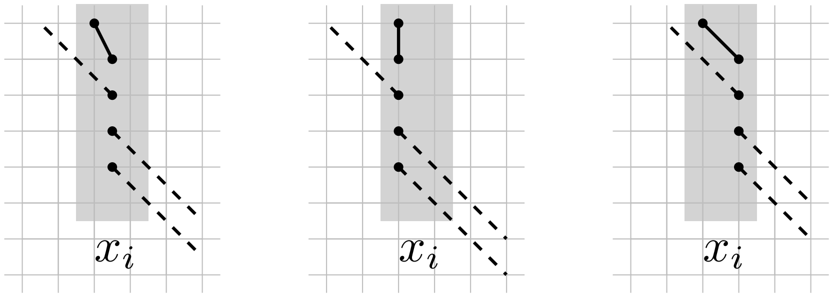
Fig. 3. The switch s is linked with edge e; (a) shows the input embedding, in (b) e is pulled up and in (c) it is pulled down.

3. if we place a vertex v with two non-integer coordinates, i.e., in the interior of a grid cell, then its position in any valid embedding is in that grid cell (including its boundary).

Basic Building Blocks. We will frequently make use of two basic building blocks that rely on the above grid properties.

The first one is a *switch*, i.e., an edge that has exactly two valid embeddings. Let $s = uv$ be an edge within a single grid cell where u is placed on a grid point and v on a non-incident grid edge. We call u the *fixed* and v the *free* vertex of s. Assume that u is in the lower left corner and v on the right edge of the grid cell. Then in any valid d-schematization s is either horizontal or diagonal, see Fig. 3 for an example.

The second basic concept is *linking* of vertices. We can synchronize two vertices in different and even non-adjacent cells of the grid by assigning them the same x- or y-coordinate. We call two vertices u and v *linked*, if in π either $x_\pi(u) = x_\pi(v)$ or $y_\pi(u) = y_p(v)$. Then the orthogonal order requires that u and v remain linked in any valid embedding. This concept allows us to transmit information on local embedding choices over distances. Two edges $e_i = uu'$ and $e_j = vv'$ are *linked* if there is a vertex of e_i that is linked to a vertex of e_j. We use linking of edges in combination with switches. Namely, we link a switch s via its free vertex with another edge e, as illustrated in Fig. 3. Then the choice of the embedding of s determines one of the two coordinates of the linked vertices in e. In the case depicted in Fig. 3 the switch s determines the y-coordinate of both vertices of e; we say that s *pulls* e up (Fig. 3(b)) or down (Fig. 3(c)). Such a switch is called a *vertical switch*. Analogously, edges can be pulled to the left and to the right by a *horizontal switch*.



(a) Input embedding (b) State `true` (c) State `false`

Fig. 4. A variable gadget with three connector vertices.

Variable Gadgets. The variable gadget for a variable x is a simple structure consisting of a horizontal switch and a number of linked *connector vertices* on consecutive grid lines below the switch, one for each appearance of x or $\neg x$ in a clause of φ. We denote the number of appearances as $t(x)$. All connector vertices share the same x-coordinate in π. Each one will be connected to a clause with a diagonal *connector edge*. A connector edge spans the same number of grid cells horizontally and vertically and hence can only be embedded at a slope of 45°. In order to have the correct slope, the positions of both endpoints within their grid cell must be the same. The upper vertices will connect to the gadgets of the negative clauses and the lower vertices to the gadgets of the positive clauses. The variable gadget has two states, one in which the switch pulls all vertices to the left (defined as `true`), and one in which it pulls them to the right (defined as `false`). Figure 4 shows an example. A variable gadget g_x for a variable x takes up one grid cell in width and $t(x)+1$ grid cells in height.

Clause Gadgets. The general idea behind the clause gadget is to create a set of paths whose embedding is influenced by three connector edges. Each of those edges carries the truth value of the connected variable gadget. The gadget consists of three diagonal edges e_2, e_3, e_4, two vertical edges e_1 and e_5 and four switches s_1, s_2, s_3, s_4, see Fig. 5. For positive clauses, we want that if all its connector edges are pulled to the right, i.e., all literals are `false`, there is no valid embedding of the clause gadget. This is achieved by placing the edges of the gadget in such a way that there are certain points, called *critical points*, where two different non-adjacent edges can possibly place a vertex. Further, we ensure with the help of switches that each of the three diagonal edges of the clause gadget has exactly one vertex embedded on a critical point in a valid embedding.

Key to the gadget is the *critical edge* e_3 that is embedded with one fixed vertex on a grid point and one loose vertex in the grid cell A to the top left of the fixed vertex. Edge e_3 is linked with switches s_2 and s_4. These switches ensure that the loose vertex must be placed on a free corner of the grid cell. The three free corners are all critical points. It is clear that there is a valid embedding of e_3 if and only if one of the three critical points is available.

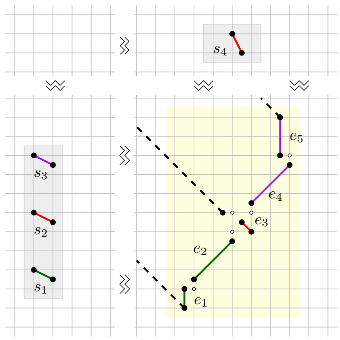


Fig. 5. Sketch of a clause gadget. Critical points are marked as white disks.

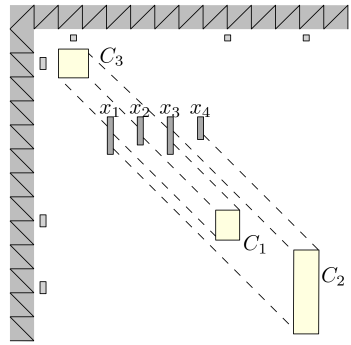


Fig. 6. Sketch of the gadgets for the instance φ from Fig.1.

We use the remaining edges of the gadget to block one of the critical points of A for each literal that is `false`. The lower vertex of the middle connector edge is placed just to the left of the upper left corner of A such that it occupies a critical point if it is pulled to the right. Both of the left and right connector edges have another linked vertical edge e_1 and e_5 appended each. Now if e_1 is pushed to the right by its connector edge, then edge e_2 is pushed upward since e_1 and e_2 share a critical point. Due to switch s_1 edge e_2 blocks the lower left critical point of A in that case. Similarly, edge e_4 blocks the upper right critical point of A if the third literal is `false`.

The clause gadget for a negative clause works analogously such that a critical point is blocked for each connector edge that is pulled to the left instead of to the right. It corresponds basically to the positive clause gadget rotated by 180°.

Lemma 2. *A clause gadget has a valid embedding if and only if at least one of its literals is* `true`*.*

The size of a clause gadget (not considering the switches) depends on the horizontal distances Δ_1, Δ_2 between the left and middle connector edge and between the middle and right connector edge, respectively. The width of a gadget is 6 and its height is $\Delta_1 + \Delta_2 - 5$.

Gadget Placement. The top and left part of our construction is occupied by the border gadget that defines the grid. The overall shape of the remaining construction is similar to a slanted version of the standard embedding of an instance of MONOTONE PLANAR 3-SAT as in Fig. 1.

We place the individual variable gadgets horizontally aligned in the center of the drawing. Since variable gadgets do not move vertically there is no danger of accidentally linking vertices of different variable gadgets by placing them at the same y-coordinates. For the clause gadgets to work as desired, we must make sure that any two connector edges to the same clause gadget have a minimum distance of seven grid cells. This can easily be achieved by spacing the variable gadgets horizontally so that connector edges of adjacent variables cannot come too close to each other.

To avoid linking between different clause gadgets or clause gadgets and variable gadgets, we place each of them in its own x- and y-interval of the grid with the negative clauses to the top left of the variables and the positive clauses to the bottom right. The switches of each clause gadget are placed at the correct positions just next to the border gadget. Figure 6 shows a sketch of the full placement of the gadgets for a MONOTONE PLANAR 3-SAT instance φ. Since the size of each gadget is polynomial in the size of the formula φ, so is the whole construction. The above construction finally yields the following theorem.

Theorem 1. *The 2-regular* UNION OF PATHS SCHEMATIZATION PROBLEM *is NP-hard.*

3.2 Hardness of the PATH SCHEMATIZATION PROBLEM

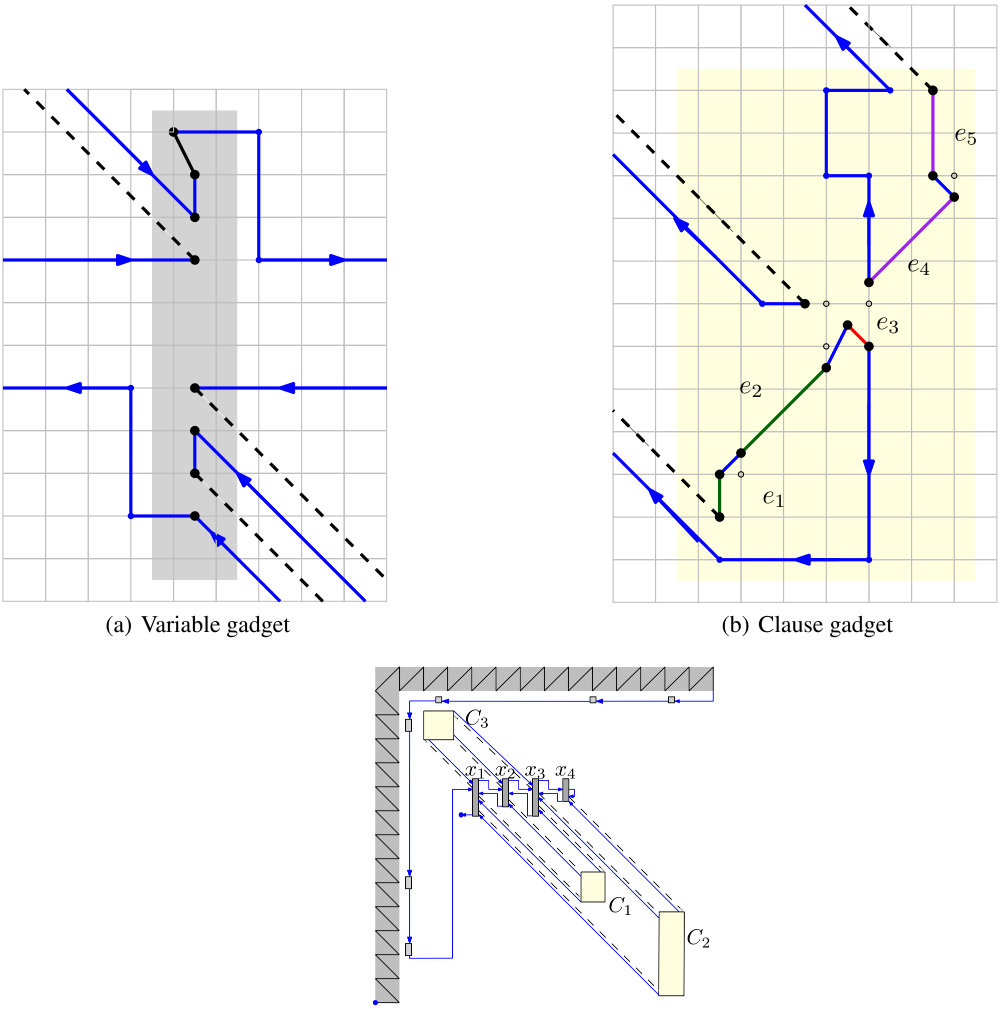
Finally, to prove that 2-regular PSP is also NP-hard we have to show that we can augment the union of paths constructed above to form a single simple path that still has the property that it has a valid embedding if and only if the corresponding MONOTONE PLANAR 3-SAT formula φ is satisfiable.

The general idea is to start the path at the lower end of the border gadget and then collect all the switches next to the border gadget. From there we enter the upper parts of the variable gadgets and walk consecutively along all the connector edges into the negative clause gadgets. Once all negative connectors have been traversed, the path continues along the positive connectors and into the positive clauses. The additional edges and vertices must be placed such that they do not interfere with any functional part of the construction.

The only major change is that we double the number of connector vertices in each variable gadget and add a parallel dummy edge for each connector edge. That way we can walk into a clause gadget along one edge and back into the variable gadget along the other edge. We also need a clear separation between the negative and positive connector vertices, i.e., the topmost positive connector vertices of all variable gadgets are assigned the same y-coordinate and we adjust the required spacing of the variable gadgets accordingly. Figure 7(a) shows an example of an augmented variable gadget with one positive and two negative connector edges.

The edges e_1–e_5 of each clause gadget are inserted into the path in between the leftmost connector edge and its dummy edge and in between the rightmost connector edge and its dummy edge, respectively. The details are illustrated in Fig. 7(b). It is clear that by this construction we obtain a single simple path P that contains all the gadgets of our previous reduction. Moreover, the additional edges are embedded such that they do not interfere with the gadgets themselves. Rather they can move along with the flexible parts without occupying grid points that are otherwise used by the gadgets. Hence we can summarize:

Theorem 2. *The 2-regular* PATH SCHEMATIZATION PROBLEM *is NP-hard.*



(a) Variable gadget

(b) Clause gadget

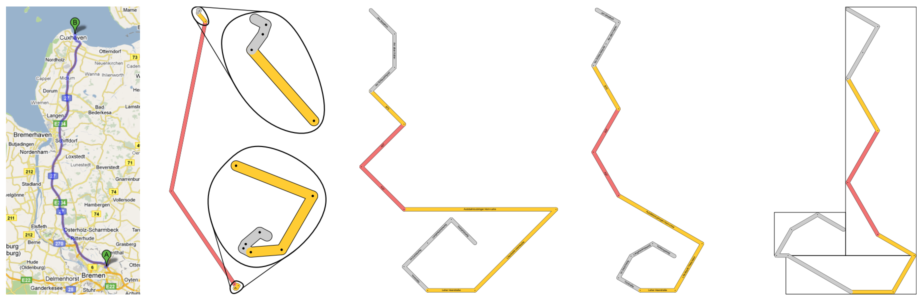
(c) Sketch of full instance

Fig. 7. Augmented gadget versions. Additional path edges are highlighted in blue.

4 Hardness of d-regular PSP for $d > 2$

In the previous section we have established that 2-regular PSP is NP-hard. Here we show that d-regular PSP remains NP-hard for all $d > 2$ by modifying the gadgets of our proof for $d = 2$.

An obvious problem for adapting the gadgets is that due to the presence of more than one diagonal slope the switches do not work any more for uniform grid cells. In a symmetric grid, we cannot ensure that the free vertex of a switch is always on a grid point in any valid d-schematization. This means that we need to devise a different grid



(a) (b) (c) (d) (e)

Fig. 8. A sample route from Bremen to Cuxhaven in Germany: (a) output of Google Maps, (b) simplified route after preprocessing, (c) output of our MIP for $d = 2$, (d) output of our MIP for $d = 3$, and (e) output of the algorithm by Delling et al. [7].

in order to make the switches work properly. We construct a new border gadget that induces a grid with cells of uniform width but non-uniform heights. We call the cells of this grid *minor cells* and form groups of $d - 1$ vertically consecutive cells to form *meta cells*. Then all meta cells again have uniform widths and heights.

The $d - 1$ different heights of the minor cells are chosen such that the segments connecting the upper left corner of a meta cell to the lower right corners of its minor cells have exactly the slopes that are multiples of $(90/d)^\circ$ and lie strictly between 0° and 90°. This restores the functionality of switches whose fixed vertex is placed on the upper left corner of a meta cell and whose free vertex is on a non-adjacent grid line of the same meta cell. Although the underlying grid differs the functionality of variable and clause gadgets as well as the overall structure stays the same. This yields the following theorem. The detailed proof is included in the full version of this paper [10].

Theorem 3. *The d-regular* PATH SCHEMATIZATION PROBLEM *is NP-hard for any* $d > 2$.

5 Design and Evaluation of a MIP for d-regular PSP

PSP can be formulated as a MIP that is similar to a MIP model for drawing octilinear metro maps [16]. With a MIP we can not only decide PSP but also optimize the output for resemblance with the input path if there is a feasible solution. We achieve this by (i) minimizing for each edge the deviation between its input slope and its output slope, and (ii) by minimizing the total path length subject to a certain minimum length for every edge. See the full version of this paper [10] for the detailed MIP formulation.

We have implemented the MIP for simple input paths and evaluate its performance by schematizing 1000 quickest routes in the German road network, where we select source and target nodes uniformly at random. The average path length is 1020.7 nodes, hence, a schematization would yield a route sketch with far too much detail. Therefore, in a preprocessing step, we simplify the path using the Douglas-Peucker algorithm [8] but keep all important nodes such as points of road and road category changes. Moreover, we shortcut self-intersecting subpaths whose lengths are below a threshold. This is to remove over- or underpasses near slip roads of highways, which are considered irrelevant for route sketches or would be depicted as an icon rather than a loop. Figure 8 illustrates an example of a route sketch obtained by our MIP. It clearly illustrates how a schematic route sketch, unlike the geographic map, is able to depict both high-detail and low-detail parts of the route in a single picture. Comparing Figures 8(c) and 8(d) indicates that using the parameter $d = 2$ for the schematization process yields route sketches with a very high level of abstraction while using $d = 3$ results in route sketches which resemble the overall shape of the route much better. For example, the route sketch in Fig. 8(c) seems to suggest that there is a 90° turn while driving on the highway but this is not the case. The route sketch depicted in Fig. 8(c) resembles the original route much better. Generally, we found that using the parameter $d = 3$ yields route sketches with a higher degree of readability.

For comparison, we show the output computed by the method of Delling et al. [7] in Fig. 8(e). Recall that their original algorithm takes as input only monotone paths. They describe, however, a heuristic approach to schematize non-monotone paths by subdividing them into maximal monotone subpaths that are subsequently merged into a single route sketch. In order to avoid intersections of the bounding boxes of the monotone subpaths, additional edges of appropriate length must be inserted into the path; the orthogonal order is preserved only within the monotone subpaths. This may lead to undesired effects in sketches of non-monotone routes, see Fig. 8(e).

Evaluation. We performed two experiments that were executed on a single core of an AMD Opteron 2218 processor running Linux 2.6.27.23. The machine is clocked at 2.6 GHz, has 16 GiB of RAM and 2×1 MiB of L2 cache. Our implementation is written in C++ and was compiled with GCC 4.3.2, using optimization level 3. As MIP solver we use Gurobi 3.0.1.

In the first experiment we examine the performance of our MIP for $d = 3$ subject to the amount of detail of the route, as controlled by the distance threshold ε between input and output path in the path simplification step. We also set the threshold for removing self-intersecting subpaths to ε. Table 1(a) reports our experimental results for values of ε between 2^{-1} and 2^{-5}.

We observe that with decreasing ε, the lengths of the paths increase from 20.04 nodes to 32.81 nodes on average. This correlates with the running time of our MIP which is between 801.61 ms and 1 102.20 ms. Note that in practice a value of $\varepsilon = 2^{-3}$ is a good compromise between computation time and amount of detail. Further, we observe that adding planarity constraints in a lazy fashion pays off since we need less than 1.3 iterations on average. constraints at all. The number of infeasible instances, i. e., paths without a valid 3-schematization, is 0.1 %. schematization respecting the orthogonal order of the nodes together with the minimum edge lengths, or, if the input path is not simple.

The second experiment evaluates the performance of our MIP when using different values of d, see Table 1(b). We fix $\varepsilon = 2^{-3}$. While for rectilinear drawings with $d = 1$ we need 107.36 ms to compute a solution, the running time increases to 1 347.86 ms

Table 1. Results obtained by running 1 000 random queries in the German road network and schematizing the simplified paths with our MIP. The tables reports average path lengths, percentage of infeasible instances, average number of iterations, and average running times.

(a) Varying ε with $d = 3$.

ε	∼length	% inf	∼iter	∼time [ms]
2^{-1}	20.04	0.1	1.28	801.61
2^{-2}	20.50	0.1	1.29	621.66
2^{-3}	21.81	0.1	1.29	649.49
2^{-4}	25.16	0.2	1.26	772.57
2^{-5}	32.81	0.3	1.27	1102.20

(b) Varying d with $\varepsilon = 2^{-3}$.

d	% inf	∼iter	∼time [ms]
1	51.3	1.03	107.36
2	0.7	1.19	363.49
3	0.1	1.29	649.49
4	0.1	1.25	1075.82
5	0.1	1.21	1347.86

when using $d = 5$. Interestingly, more than half of the paths do not have a valid rectilinear schematization. By allowing one additional diagonal slope ($d = 2$), the number of infeasible instances significantly decreases to 0.7 % and for $d = 3$ only 0.1 %, i. e., a single instance, is infeasible.

6 Conclusion

Motivated by drawing route sketches in road networks, we studied the d-regular path schematization problem (PSP), which has two main goals: To preserve the user's mental map through maintaining the orthogonal order, and to reduce the visual complexity using restricted edge slopes given by integer multiples of $(90/d)^\circ$. We analyzed the complexity of the problem and showed that PSP is NP-hard for $d \geq 2$, thus, closing the complexity gap between the hardness result of Brandes and Pampel [4] for $d = 1$, and the efficient algorithm of Delling et al. [7] for monotone paths. In the second part of this work, we modeled and implemented the PSP as a mixed integer linear program (MIP). An experimental evaluation with real-world data of the German road network showed that we are indeed able to compute visually appealing route sketches within approximately one second for values of $d \leq 5$.

Using ideas of Nöllenburg and Wolff [16], our MIP can be further generalized to handle both non-simple paths and general graphs, e. g., a set of alternative routes.

Acknowledgments. We thank an anonymous referee for helpful comments.

References

1. M. Agrawala and C. Stolte. Rendering effective route maps: Improving usability through generalization. *Proc. 28th Ann. Conf. Computer Graphics and Interactive Techniques (SIGGRAPH'01)*, pp. 241–249. ACM, 2001.
2. M. de Berg and A. Khosravi. Finding perfect auto-partitions is NP-hard. *Proc. 25th European Workshop Comput. Geom. (EuroCG'08)*, pp. 255–258, 2008.
3. U. Brandes, M. Eiglsperger, M. Kaufmann, and D. Wagner. Sketch-driven orthogonal graph drawing. *Proc. 10th International Symposium on Graph Drawing (GD'02)*, pp. 1–11. Springer-Verlag, Lecture Notes Comput. Sci. 2528, 2002.

4. U. Brandes and B. Pampel. On the hardness of orthogonal-order preserving graph drawing. *Proc. 16th Internat. Symp. Graph Drawing (GD'08)*, pp. 266–277. Springer-Verlag, Lecture Notes Comput. Sci. 5417, 2009.
5. J. Branke. Dynamic graph drawing. *Drawing Graphs: Methods and Models*, chapter 9, pp. 228–246. Springer-Verlag, Lecture Notes Comput. Sci. 2025, 2001.
6. S. Cabello, M. d. Berg, S. v. Dijk, M. v. Kreveld, and T. Strijk. Schematization of road networks. *Proc. 17th Annual Symposium on Computational Geometry (SoCG'01)*, pp. 33–39. ACM Press, 3–5 June 2001.
7. D. Delling, A. Gemsa, M. Nöllenburg, and T. Pajor. Path schematization for route sketches. *Proc. 12th Scand. Symp. & Workshops on Algorithm Theory (SWAT'10)*, pp. 285–296. Springer-Verlag, Lecture Notes Comput. Sci. 6139, 2010.
8. D. H. Douglas and T. K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica* 10(2):112–122, 1973.
9. T. Dwyer, Y. Koren, and K. Marriott. Stress majorization with orthogonal ordering constraints. *Proc. 13th Internat. Symp. Graph Drawing (GD'05)*, pp. 141–152. Springer-Verlag, Lecture Notes Comput. Sci. 3843, 2006.
10. A. Gemsa, M. Nöllenburg, T. Pajor, and I. Rutter. On d-regular Schematization of Embedded Paths. Tech. Rep. 2010-21, Faculty of Informatics, Karlsruhe Institute of Technology, 2010. `http://digbib.ubka.uni-karlsruhe.de/volltexte/1000020426`
11. K. Hayashi, M. Inoue, T. Masuzawa, and H. Fujiwara. A layout adjustment problem for disjoint rectangles preserving orthogonal order. *Proc. 6th Internat. Symp. Graph Drawing (GD'98)*, pp. 183–197. Springer-Verlag, Lecture Notes Comput. Sci. 1547, 1998.
12. D. Merrick and J. Gudmundsson. Path simplification for metro map layout. *Proc. 14th Internat. Symp. Graph Drawing (GD'06)*, pp. 258–269. Springer-Verlag, Lecture Notes Comput. Sci. 4372, 2007.
13. K. Misue, P. Eades, W. Lai, and K. Sugiyama. Layout adjustment and the mental map. *J. Visual Languages and Computing* 6(2):183–210, 1995.
14. G. Neyer. Line simplification with restricted orientations. *Proc. 6th Int. Workshop on Algorithms and Data Structures (WADS'99)*, pp. 13–24. Springer-Verlag, Lecture Notes Comput. Sci. 1663, 1999.
15. M. Nöllenburg. Automated drawing of metro maps. Tech. Rep. 2005-25, Fakultät für Informatik, Universität Karlsruhe, 2005, `http://digbib.ubka.uni-karlsruhe.de/volltexte/1000004123`.
16. M. Nöllenburg and A. Wolff. Drawing and Labeling High-Quality Metro Maps by Mixed-Integer Programming. *IEEE Transactions on Visualization and Computer Graphics*, 2010, `http://dx.doi.org/10.1109/TVCG.2010.81`. Preprint available online.
17. A. Wolff. Drawing subway maps: A survey. *Informatik - Forschung und Entwicklung* 22(1):23–44, 2007.