Path Schematization for Route Sketches

Autoren: Daniel Delling, Andreas Gemsa, Martin Nöllenburg, Thomas Pajor

Interner Bericht 2010-02
Abstract. Motivated from drawing route sketches, we consider the following problem. We are given a simple embedded polygonal path $P = (v_1, \ldots, v_n)$, consisting of $k$ $x$- or $y$-monotone subpaths, and a set $C$ of admissible edge orientations including the coordinate axes. The problem is to redraw $P$ schematically such that all edges are drawn as line segments that are parallel to one of the specified orientations. We also require that the path preserves the orthogonal order, at least for each monotone subpath, and that it remains intersection-free. Finally, we want the drawing to be cost-minimal, i.e., the number of edges of $P$ that are not drawn according to their preferred orientation in $C$ is minimized. Moreover, the length of each edge does not fall below a given minimum length.

In this paper we present a four-step approach to this problem. We first split the input path in $O(n)$ time into $k$ monotone subpaths. Then, we embed each subpath individually in $O(n^2)$ time. For this embedding we optimize the edge lengths in a third step using a linear program. Finally, we give an $O(k^2 + n)$-time algorithm to combine all $k$ schematic embeddings of the monotone subpaths into a single, intersection-free embedding of the initial path $P$.

1 Introduction

Simplification and schematization of map objects are well-known operators in cartographic generalization, i.e., the process to adapt map content to its scale and use. Simplification usually reduces unnecessary complexity, e.g., by removing extraneous vertices of a polygonal line while still maintaining its overall appearance. Schematization, however, may abstract more drastically from geographic reality as long as the intended map use allows for it. Public transport maps are good examples of schematization, where edge orientations are limited to a small number of slopes and edge lengths are no longer drawn to scale [9]. In spite of all distortions, such maps usually work well.

In this paper we consider a path schematization problem that is motivated from visualizing routes in road networks. Routes typically begin and end in residential or commercial areas, where roads are mostly used only for short distances of a few meters up to a few hundred meters. As soon as the route leaves the city limits, however, country roads and highways tend to be used for distances ranging from a few up to hundreds of kilometers. Moreover, optimal routes tend to follow a general driving direction and deviations from this direction are rare.

Commercial route planners typically present driving directions for such routes as a graphical overview of the route highlighted in a traditional road map (see Fig. 1a) in combination with a textual step-by-step description. The overview map is good for giving a general idea of the route, but due to its small scale it often does not succeed in showing details of the route, in particular for short roads in the vicinity of start and destination and off the main highways. Textual descriptions are accurate when used at the right moment but there is a high risk of loss of context. On the other hand, a manually drawn route sketch often shows the whole route in a single picture, where
each part of the route has its own appropriate scale: important turning points along the route and short residential roads are enlarged while long stretches of highways and country roads are shortened. Edges are often aligned with a small set of orientations rather than being geographically accurate [11]. Figure 1b gives an example. In spite of the cartographic error, such route sketches are often easier to read than textual descriptions and traditional road maps—at least if the user’s mental or cognitive map, i.e., a rough idea of the geographic reality, is preserved [7,10].

We formalize the application problem of drawing route sketches as a geometric path schematization problem. Given a plane embedding of a path $P$, the goal is to find a schematic embedding of $P$ that is as similar to the input embedding as possible but uses only a restricted set $C$ of edge orientations and has no edges shorter than a fixed minimum length. An embedding $\pi$ of $P$ is denoted $C$-oriented if each edge in $\pi$ is parallel to some orientation $\gamma \in C$. For our application of route sketches, the path $P$ is given by $n$ important points along the route. These important points can be turns, important junctions, highway ramps, etc.

Related Work. Similar path schematization problems have been studied before. Neyer [8] proposed an algorithm to solve the $C$-oriented line simplification problem, where a $C$-oriented simplification $Q$ of a polygonal path $P$ is to be computed that uses a minimum number of edges. Furthermore, $Q$ must have Fréchet distance at most $\varepsilon$ from $P$. For a constant-size set $C$ the algorithm has a running time of $O(kn^2 \log n)$, where $n$ is the number of vertices of $P$ and $k$ is the number of vertices of $Q$. Merrick and Guinandsson [6] studied a slightly relaxed version of the same problem and gave an $O(n^2 |C|^3)$-time algorithm to compute a $C$-oriented simplification of $P$ that is within Hausdorff distance at most $\varepsilon$ of $P$. Agrawala and Stolte [1] designed a system called LineDrive that uses heuristic methods based on simulated annealing in order to render route maps similar to hand-drawn sketches. While their system allows distortion of edge lengths and angles, the resulting paths are neither $C$-oriented nor can hard quality guarantees be given. They did, however, implement and evaluate the system in a study that showed that users generally preferred LineDrive route maps over traditional route maps. Brandes and Pampel [2] studied the path schematization problem in the presence of orthogonal order constraints [7] in order to preserve the mental map. They showed that
deciding whether a rectilinear schematization of a path \( P \) exists that preserves the orthogonal order of \( P \) is NP-hard. They also showed that schematizing a path using arbitrarily oriented unit-length edges is NP-hard.

**Our Contribution.** We present solutions to the following four algorithmic subproblems for drawing route sketches. Their combination yields an efficient method to compute schematic route sketches.

- The first (rather simple) problem is to split the input path into a minimum number \( k \) of \( x \)- or \( y \)-monotone subpaths; we present a straight-forward linear-time algorithm (Section 3.1).
- The second problem is the monotone path schematization problem (Section 3.2), where we are given an \( x \)- or \( y \)-monotone embedding \( \pi \) of a path \( P \) with \( n \) vertices and a set \( C \) of edge orientations. We present an \( O(n^2) \)-time algorithm to compute a \( C \)-oriented embedding \( \rho \) of \( P \) that preserves the orthogonal order of \( \pi \) and minimizes the number of edges not drawn in the direction of their closest approximation in \( C \). Note that the orthogonal order of graph drawings has been used before as a means to preserve the mental map \([2,4,7]\).
- The third problem is the total edge length minimization problem (Section 3.3), where the aim is to minimize the sum of all edge lengths such that all edge directions and the orthogonal order of \( \rho \) are preserved and additionally each edge is no shorter than a given minimum length. We present a linear program for this problem, which, in our case, can be solved in \( O(n^3 \log^2 n) \) time.
- Finally, we propose an \( O(k^2 + n) \)-time algorithm to combine \( k \) monotone subpaths into a single intersection-free path with \( O(n) \) vertices (Section 3.4).

In our opinion, the most relevant subproblems for drawing route sketches are problems 2 and 3 as for route sketches, the input path is mostly monotone and \( k \) tends to be small (see the example in Fig. 1c). Moreover, rectilinear path schematization, which is a special case of \( C \)-oriented path schematization, is NP-hard for general paths \([2]\); this further motivates studying the schematization of monotone paths.

## 2 Preliminaries

Let \( P = (v_1, \ldots, v_n) \) be a path with edges \( v_iv_{i+1} \) for \( 1 \leq i \leq n - 1 \). For a vertex \( v \) and an edge \( e \) of \( P \) we say \( v \in P \) and \( e \in P \). A **plane embedding** \( \pi : P \rightarrow \mathbb{R}^2 \) maps each vertex \( v_i \in P \) to a point \( \pi(v_i) = (x_\pi(v_i), y_\pi(v_i)) \) and each edge \( uv \in P \) to the line segment \( \pi(uv) = [\pi(u), \pi(v)] \) such that \( \pi \) is a simple polygonal path with vertex set \( \{\pi(v_1), \ldots, \pi(v_n)\} \). We denote the length of an edge \( e \) in \( \pi \) as \( |\pi(e)| \). An **embedded path** is a pair \((P, \pi)\) of a path \( P \) and a plane embedding \( \pi \) of \( P \).

Let \( C = \{\gamma_1, \ldots, \gamma_k\} \) be a set of angles w.r.t. the \( x \)-axis that represents the admissible edge orientations. We require that \( \{0^\circ, 90^\circ, 180^\circ, 270^\circ\} \subseteq C \). Reasonable sets of edge directions for route sketches are, e.g., multiples of 30 or 45 degrees. Recall that a plane embedding \( \pi \) of a path is called **\( C \)-oriented** if the direction of each edge in \( \pi \) corresponds to an angle in \( C \). For an embedding \( \pi \) of \( P \) and an edge \( e \in P \) we denote by \( \alpha_{\pi}(e) \) the angle of \( \pi(e) \) w.r.t. the \( x \)-axis. For the input embedding \( \pi \), we similarly denote by \( \omega_C(e) \) the **preferred angle** \( \gamma \in C \), i.e., the angle in \( C \) that is closest to \( \alpha_{\pi}(e) \). For a \( C \)-oriented embedding \( \rho \) of \( P \) and an edge \( e \in P \) there is a **direction cost** \( c_{\rho}(e) \) that captures how much the angle \( \alpha_{\rho}(e) \) deviates from \( \omega_C(e) \). The **schematization cost** \( c(\rho) \) is then defined as \( c(\rho) = \sum_{e \in P} c_{\rho}(e) \).

Following Misue et al. \([7]\), we say that an embedding \( \rho \) of a path \( P \) preserves the **orthogonal order** of another embedding \( \pi \) of \( P \) if for any two vertices \( v_i \) and \( v_j \in P \) we have \( x_{\pi}(v_i) \leq x_{\pi}(v_j) \) if
and only if \( x_\rho(v_i) \leq x_\rho(v_j) \) and \( y_\pi(v_i) \leq y_\pi(v_j) \) if and only if \( y_\rho(v_i) \leq y_\rho(v_j) \). In other words, any two vertices keep their above-below and left-right relationship.

Finally, a \textit{monotone path decomposition} for an embedded path \((P, \pi)\) is a decomposition \( \mathcal{P} = \{(P_1, \pi_1), \ldots, (P_k, \pi_k)\} \) of \((P, \pi)\) into consecutive \(x\)- or \(y\)-monotone embedded subpaths \((P_i, \pi_i)\) for \(1 \leq i \leq k\), where \(\pi_i\) equals \(\pi\) restricted to the subpath \(P_i\).

## 3 Path Schematization

Our approach to schematizing a path consists of four steps. First, we split the path into a minimal number of \(x\)- or \(y\)-monotone subpaths. Afterwards, we compute an orthogonal-order preserving \(C\)-oriented schematization of each monotone subpath that minimizes the schematization cost. Since edge lengths are not optimized yet, we formulate a linear program that optimizes the edge lengths of each embedding obtained in step 2 without modifying edge orientations or the orthogonal order. Our final step merges all subpaths into the final schematization of the input path.

### 3.1 Step 1: Splitting the Input

In a first step, we divide the path \(P = (v_1, \ldots, v_n)\) into the minimal number \(k\) of subpaths \(P_i, 1 \leq i \leq k\) with the property that each \(P_i\) is an \(x\)- or \(y\)-monotone path. This can be done in greedy fashion, starting from \(v_1\). We traverse \(P\) until we find the first vertices \(v'\) and \(v''\) violating the \(x\)- and \(y\)-monotonicity, respectively. If \(v'\) appears later than \(v''\) on \(P\), we set \(P_1 = (v_1, \ldots, v')\), otherwise \(P_1 = (v_1, \ldots, v'')\). We continue this procedure until we reach the end of \(P\). This algorithm runs in \(O(n)\) time and returns the minimal number \(k\) of \(x\)- or \(y\)-monotone subpaths, indicated by Proposition 1.

**Proposition 1.** The greedy path splitting algorithm divides the input path \(P\) into a minimal number of \(x\)- or \(y\)-monotone subpaths in linear time.

**Proof.** Assume \((L_1, \ldots, L_l)\) is an optimal solution that divides \(P\) into \(l\) \(x\)- or \(y\)-monotone subpaths, and let \((Q_1, \ldots, Q_k)\) be the solution obtained by the above algorithm. Observe that due to the greedy approach the following two statements hold: (i) the path \(L_i\) for the smallest \(i\) such that \(L_i\) differs from \(Q_i\) contains less vertices than \(Q_i\); (ii) there cannot be any path \(L_m\) that fully contains some path \(Q_j = (v_p, \ldots, v_q)\) and also \(v_{q+1} \in L_m\). This implies that if there is a path \(L_m\) that contains \(Q_j\) they both share the same last vertex. Combined, we get that there is no sequence of paths \(L_1, \ldots, L_i\) containing more vertices than the sequence \(Q_1, \ldots, Q_i\) and hence \(k \leq l\). \(\Box\)

After splitting the path \(P\), we assign a preferred angle \(\omega_C(e)\) to each edge \(e \in P\). In general, this is the angle \(\gamma \in \mathcal{C}\) closest to \(\alpha_\pi(e)\). This could, however, result in the following conflict. Consider two subsequent edges \(e_1, e_2\) with \(\omega_C(e_1) = (\omega_C(e_2) + 180^\circ)\mod 360^\circ\). Assigning such preferred angles would result in an overlap of \(e_1\) and \(e_2\). Hence, in such a case, we set either \(\omega_C(e_1)\) or \(\omega_C(e_2)\) to its next best value, depending on which edge is closer to its next best value. Note that assigning \(\omega_C(e)\) to each \(e\) can be done in \(O(n)\) time.

### 3.2 Step 2: Orthogonal-Order Preserving Schematization of Monotone Subpaths

Having computed a monotone path decomposition \(\mathcal{P}\) of the embedded input path \((P, \pi)\), our aim is now to schematize each embedded \(x\)- or \(y\)-monotone subpath \((Q, \pi) \in \mathcal{P}\). Here, we restrict ourselves to the case of \(x\)-monotone paths; \(y\)-monotone paths are schematized analogously.
Problem 1. Given an embedded x-monotone path \((Q, \pi)\) and a set \(C\) of edge orientations, find a plane \(C\)-oriented embedding \(\rho\) of \(Q\) that is orthogonal-order preserving and minimizes the schematization cost \(c(\rho)\).

We first observe that \(\rho\) is also x-monotone as it preserves the orthogonal order of \(\pi\). So we can assume that \(Q = (v_1, \ldots, v_n)\) is ordered from left to right in both embeddings. Let \(\rho'\) be any orthogonal-order preserving embedding of \(Q\). Then we can always modify the \(x\)-coordinates of \(\rho'\) such that any edge \(e = v_iv_{i+1}\) of \(Q\) with \(\omega(\rho')(e) \neq 0^\circ\) and \(y_{\rho'}(v_i) \neq y_{\rho'}(v_{i+1})\) satisfies \(\alpha_{\rho'}(e) = \omega(\rho')(e)\). This is done by horizontally shifting the whole embedding \(\rho'\) right of \(x_{\rho'}(v_{i+1})\) until the slope of \(e\) satisfies \(\alpha_{\rho'}(e) = \omega(\rho')(e)\). Due to the x-monotonicity no other edges are affected by this shift.

Next, we group all edges \(e = uv\) of \(Q\) into four categories according to their preferred angle \(\omega(\rho')(e)\):

1. if \(\omega(\rho')(e) = 0^\circ\) and \(y_{\rho'}(u) \neq y_{\rho'}(v)\) then \(e\) is called horizontal edge (or h-edge);
2. if \(y_{\rho'}(u) = y_{\rho'}(v)\) then \(e\) is called strictly horizontal edge (or sh-edge);
3. if \(\omega(\rho')(e) \neq 0^\circ\) and \(x_{\rho'}(u) \neq x_{\rho'}(v)\) then \(e\) is called vertical edge (or v-edge);
4. if \(x_{\rho'}(u) = x_{\rho'}(v)\) then \(e\) is called strictly vertical edge (or sv-edge).

We define a binary direction cost as follows. All edges \(e\) with \(\alpha_{\rho}(e) = \omega(\rho')(e)\) are drawn according to their preferred angle and we assign the cost \(c_{\rho}(e) = 0\). For all edges \(e\) with \(\alpha_{\rho}(e) \neq \omega(\rho')(e)\) we assign the cost \(c_{\rho}(e) = 1\). An exception are the sh- and sv-edges, which must be assigned their preferred angle due to the orthogonal ordering constraints. Consequently, we set \(c_{\rho}(e) = \infty\) for any sh- or sv-edge \(e\) with \(\alpha_{\rho}(e) \neq \omega(\rho')(e)\). Using the above horizontal shifting argument, the cost \(c_{\rho}(e)\) of any edge \(e\) depends only on the vertical distance between its endpoints. Hence, the schematization cost of an x-monotone embedding \(\rho\) is already determined by assigning \(y\)-coordinates \(y_{\rho}(v)\) to all vertices \(v\) of \(Q\). Note that our algorithm can be easily adapted to alternative cost definitions.

To that end, we define \(m \leq n - 1 \) closed and vertically bounded horizontal strips \(s_1, \ldots, s_m\) induced by the set \(\{y = y_{\rho}(v_i) \mid 1 \leq i \leq n\}\) of horizontal lines through the vertices of \((Q, \pi)\). Let these strips be ordered from top to bottom as shown in Fig. 2a. Furthermore we define a dummy strip \(s_0\) above \(s_1\) that is unbounded on its upper side. We say that an edge \(e = uv\) crosses a strip \(s_i\) and conversely that \(s_i\) affects \(e\) if \(\pi(u)\) and \(\pi(v)\) lie on opposite sides of \(s_i\). In fact, to determine the cost of an embedding \(\rho\) it is enough to know for each strip whether it has a positive height or not. Our algorithm will assign a symbolic height \(h(s_i) \in \{0, 1\}\) to each strip \(s_i\) such that the schematization cost is minimum. Note that sh-edges do not cross any strip but rather coincide with some strip boundary. Hence all sh-edges are automatically drawn horizontally and have no direction costs. We can therefore assume that there are no sh-edges in \((Q, \pi)\).

Let \(S[i, j] = \bigcup_{k=i}^{j} s_k\) be the union of the strips \(s_1, \ldots, s_j\) and let \(\mathcal{I}(i, j)\) be the subinstance of the path schematization problem containing all edges that lie completely within \(S[i, j]\). Note that \(\mathcal{I}(1, m)\) corresponds to the original instance \((Q, \pi)\), whereas in general \(\mathcal{I}(i, j)\) is no longer a connected path but a collection of edges. The following lemma is a key to our algorithm.

Lemma 1. Let \(\mathcal{I}(i, j)\) be a subinstance of the path schematization problem and let \(s_k \subseteq S[i, j]\) be a strip for some \(i \leq k \leq j\). If we assign \(h(s_k) = 1\) then \(\mathcal{I}(i, j)\) decomposes into the two independent subinstances \(\mathcal{I}(i, k-1)\) and \(\mathcal{I}(k+1, j)\). The direction costs of all edges affected by \(s_k\) are determined by setting \(h(s_k) = 1\).

Proof. We first show that the cost of any edge \(e = uv\) that crosses \(s_k\) is determined by setting \(h(s_k) = 1\). Since \(u\) and \(v\) lie on opposite sides of \(s_k\) we know that \(y_{\rho}(u) \neq y_{\rho}(v)\). So if \(e\) is a v- or
sv-edge it can be drawn with its preferred angle and \( c_\rho(e) = 0 \) regardless of the height of any other strip crossed by \( e \). Conversely, if \( e \) is an h-edge it is impossible to draw \( e \) horizontally regardless of the height of any other strip crossed by \( e \) and \( c_\rho(e) = 1 \). Recall that sh-edges do not cross any strips. Assume that \( k = 2 \) in Fig. 2a and we set \( h(s_2) = 1 \); then edges \( v_3v_4 \) and \( v_5v_6 \) cross strip \( s_2 \) and none of them can be drawn horizontally.

The remaining edges of \( \mathcal{I}(i, j) \) do not cross \( s_k \) and are either completely contained in \( S[i, k - 1] \) or in \( S[k + 1, j] \). Since the costs of all edges affected by \( s_k \) are independent of the heights of the remaining strips in \( S[i, j] \setminus \{s_k\} \), we can solve the two subinstances \( \mathcal{I}(i, k - 1) \) and \( \mathcal{I}(k + 1, j) \) independently, see Fig. 2a.\footnote{We can now describe our algorithm for assigning symbolic heights to all strips \( s_1, \ldots, s_m \) such that the induced embedding \( \rho \) has minimum schematization cost. The main idea is to recursively compute an optimal solution for each instance \( \mathcal{I}(1, i) \) by finding the best \( k \leq i \) such that \( h(s_k) = 1 \) and \( h(s_j) = 0 \) for \( j = k + 1, \ldots, i \). By using dynamic programming we can compute an optimal solution for \( \mathcal{I}(1, m) = (Q, \pi) \) in \( O(n^2) \) time.

Let \( C(k, i) \) for \( 1 \leq k \leq i \) denote the schematization cost of all edges in the instance \( \mathcal{I}(1, i) \) that either cross \( s_k \) or have both endpoints in \( S[k + 1, i] \) if we set \( h(s_k) = 1 \) and \( h(s_j) = 0 \) for \( j = k + 1, \ldots, i \). Let \( C(0, i) \) denote the schematization cost of all edges in the instance \( \mathcal{I}(1, i) \) if \( h(s_j) = 0 \) for all \( j = 1, \ldots, i \). We use an array \( T \) of size \( m + 2 \) to store the minimum schematization cost \( T[i] \) of the instance \( \mathcal{I}(1, i) \). Then \( T[i] \) is recursively defined as follows

\[
T[i] = \begin{cases} 
\min_{0 \leq k \leq i} (T[k - 1] + C(k, i)) & \text{if } 1 \leq i \leq m \\
0 & \text{if } i = 0 \text{ or } i = -1.
\end{cases}
\]

Together with \( T[i] \) we store the index \( k \) that achieves the minimum value in the recursive definition of \( T[i] \) as \( k[i] = k \). This will later allow us to compute the actual strip heights using backtracking.

Note that \( T[m] < \infty \) since, e.g., the solution that assigns height 1 to every strip induces cost 0 for all sv-edges. Obviously, we need \( O(m) \) time to compute each entry in \( T \) assuming that the schematization cost \( C(k, i) \) is available in \( O(1) \) time. This yields a total running time of \( O(m^2) \).

The next step is to precompute the schematization cost \( C(k, i) \) for any \( 0 \leq k \leq i \leq m \). This cost is composed of two parts. The first part is the schematization cost of all edges that are affected by \( s_k \). As observed in Lemma 1, all v- and sv-edges crossing \( s_k \) have no cost. On the other hand, every h-edge that crosses \( s_k \) has cost 1. So we need to count all h-edges in \( \mathcal{I}(1, i) \) that
cross $s_k$. The second part is the cost of all edges that are completely contained in $S[k+1,i]$. Since $h(s_{k+1}) = \ldots = h(s_i) = 0$ we observe that any h-edge in $S[k+1,i]$ is drawn horizontally at no cost. In contrast, no v- or sv-edge $e$ in $S[k+1,i]$ attains its preferred angle $\omega(e) \neq 0^\circ$. Hence every v-edge in $S[k+1,i]$ has cost 1 and every sv-edge has cost $\infty$. So we need to check whether there is an sv-edge contained in $S[k+1,i]$ and if this is not the case count all v-edges contained in $S[k+1,i]$.

In order to efficiently compute the values $C(k,i)$ we assign to each strip $s_i$ three sets of edges. Let $H(i)$ (resp. $V(i)$ or $SV(i)$) be the set of all h-edges (resp. v-edges or sv-edges) whose lower endpoint lies on the lower boundary of $s_i$. We can compute $H(i)$, $V(i)$, and $SV(i)$ in $O(n)$ time for all strips $s_i$. Then for $k \leq i$ the number of h-edges in $H(i)$ that cross $s_k$ is denoted by $\sigma_H(k,i)$ and the number of v-edges in $V(i)$ that do not cross $s_k$ is denoted by $\sigma_V(k,i)$. Finally, let $\sigma_{SV}(k,i)$ be the number of sv-edges in $SV(i)$ that do not cross $s_k$.

This allows us to compute the values $C(k,i)$, $0 \leq k \leq i \leq m$, recursively as follows

$$C(k,i) = \begin{cases} 
\infty & \text{if } \sigma_{SV}(k,i) \geq 1 \\
C(k,i-1) + \sigma_H(k,i) + \sigma_V(k,i) & \text{if } k \leq i-1 \\
\sigma_H(k,k) & \text{if } k = i.
\end{cases}$$

(2)

Since each edge appears in exactly one of the sets $H(i)$, $V(i)$, or $SV(i)$ for some $i$ it is counted towards at most $m$ values $\sigma_H(\cdot,i)$, $\sigma_V(\cdot,i)$, or $\sigma_{SV}(\cdot,i)$, respectively. Thus for computing all these values we need $O(nm)$ time. The values $C(k,i)$ can be precomputed in $O(m^2)$ time and require a table of size $O(m^2)$. This can be reduced, however, to $O(m)$ space as follows. We compute and store the values $T[i]$ in the order $i = 1, \ldots, m$. For computing the entry $T[i]$ we use only the values $C(\cdot,i)$. To compute the next entry $T[i+1]$ we first compute the values $C(\cdot,i+1)$ from $C(\cdot,i)$ and then discard all $C(\cdot,i)$. This reduces the required space to $O(m)$. Since $m \leq n$ we obtain

**Theorem 1.** Our algorithm to compute the array $T$ of path schematization costs requires $O(n^2)$ time and $O(n)$ space.

It remains to determine the strip height assignments corresponding to the schematization cost in $T[m]$ and show the optimality of that solution. We initialize all heights $h(s_i) = 0$ for $i = 1, \ldots, m$. Recall that $k[i]$ equals the index $k$ that minimized the value $T[i]$ in (1). To find all strips with height 1 we initially set $j = m$. If $k[j] = 0$ we stop; otherwise we assign $h(s_{k[j]}) = 1$, update $j = k[j] - 1$, and continue with the next index $k[j]$ until we hit $k[j] = 0$ for some $j$ encountered in this process. Let $\rho$ be the embedding of $Q$ induced by this strip height assignment, see Fig. 2b. We now show the optimality of $\rho$ in terms of the schematization cost.

**Theorem 2.** Given an x-monotone embedded path $(Q, \pi)$ and a set $C$ of edge orientations, our algorithm computes a plane $C$-oriented embedding $\rho$ of $Q$ that preserves the orthogonal order of $\pi$ and has minimum schematization cost $c(\rho)$.

**Proof.** Since the path is x-monotone and by construction there are no two adjacent edges with preferred angles $90^\circ$ and $270^\circ$ (see Section 3.1) the embedding $\rho$ is plane. Furthermore, the $x$-coordinates of the vertices are chosen such that $\rho$ is $C$-oriented, see Fig. 2b. The embedding $\rho$ also preserves the orthogonal order of $\pi$ since $\rho$ does not alter the $x$- and $y$-ordering of the vertices of $Q$.

We show that $\rho$ has minimum schematization cost by structural induction. For an instance with a single strip $s$ there are only two possible solutions of which our algorithm chooses the better
one. The induction hypothesis is that our algorithm finds an optimal solution for any instance with at most \( m \) strips. So let’s consider an instance with \( m + 1 \) strips and let \( \rho' \) be any optimal plane \( C \)-oriented and orthogonal-order preserving solution for this instance. If all strips \( s \) in \( \rho' \) have height \( h(s) = 0 \) then by (1) it holds that \( c(\rho) = T[m + 1] \leq C(0, m + 1) = c(\rho') \). Otherwise, let \( k \) be the largest index for which \( h(s_k) = 1 \) in \( \rho' \). When computing \( T[m + 1] \) our algorithm also considers the case where \( s_k \) is the bottommost strip of height 1, which has a cost of \( T[k - 1] + C(k, m + 1) \). If \( h(s_k) = 1 \) we can split the instance into two independent subinstances to both sides of \( s_k \) by Lemma 1. The schematization cost \( C(k, m + 1) \) contains the cost for all edges that cross \( s_k \) and this cost is obviously the same as in \( \rho' \) since \( h(s_k) = 1 \) in both embeddings. Furthermore, \( C(k, m + 1) \) contains the cost of all edges in the subinstance below \( s_k \), for which we have by definition \( h(s_{k+1}) = \ldots = h(s_{m+1}) = 0 \). Since \( k \) is the largest index with \( h(s_k) = 1 \) in \( \rho' \) this is also exactly the same cost that this subinstance has in \( \rho' \). Finally, the independent subinstance above \( s_k \) has at most \( m \) strips and hence \( T[k - 1] \) is the minimum cost for this subinstance by induction. It follows that \( c(\rho) = T[m + 1] \leq T[k - 1] + C(k, m + 1) \leq c(\rho') \). This concludes the proof.

### 3.3 Step 3: Optimizing Edge Lengths

In the previous step we obtained a \( C \)-oriented and orthogonal-order preserving embedding \( \rho \) with minimum schematization cost for an embedded input path \((Q, \pi)\). The strip heights assigned in the previous step are either 0 or 1 and this does not take into account the actual edge lengths induced by \( \rho \). For drawing route sketches, however, there are certain length constraints for the edges in a schematized path. On the one hand, edges must not be shorter than some minimum length in order to remain distinguishable. On the other hand, edges that are relatively long in \( \pi \) shall not shrink too drastically during schematization. In practice, many choices for the minimum length of an edge are conceivable; here we simply assume that for each edge \( e \) of \( Q \) a minimum length \( \ell_{\min}(e) \) is given.

**Problem 2.** Given an embedded \( C \)-oriented \( x \)-monotone path \((Q, \rho)\) and a minimum edge length \( \ell_{\min}(e) \) for each edge \( e \in Q \), find a plane \( C \)-oriented embedding \( \zeta \) of \( Q \) such that

- (i) \( \zeta \) preserves the orthogonal order of \( \rho \),
- (ii) \( \zeta \) preserves the edge orientations of \( \rho \), i.e., for every edge \( e \) of \( Q \) it holds that \( \alpha_C(e) = \alpha_\rho(e) \),
- (iii) \( \zeta \) respects the minimum lengths, i.e., for every edge \( e \) it holds that \( |\zeta(e)| \geq \ell_{\min}(e) \), and
- (iv) \( \zeta \) minimizes the total stretching cost \( c_s(\zeta) := \sum_{e \in Q} (|\zeta(e)| - \ell_{\min}(e)) \).

Note that we can immediately assign the minimum length to every horizontal edge \( e \) in the input \((Q, \rho)\) by horizontally shifting the subpaths to both sides of \( e \). For any non-horizontal edge \( e = uv \) the length \( |\zeta(e)| \) depends only on the vertical distance \( \Delta_y(e) = |y_\zeta(u) - y_\zeta(v)| \) and the angle \( \alpha_C(e) \). In fact, \( |\zeta(e)| = \Delta_y(e)/\sin \alpha_C(e) \). So in order to solve Problem 2 we need to find \( y \)-coordinates for all strip boundaries such that \( c_s(\zeta) \) is minimized. These \( y \)-coordinates together with the given angles \( \alpha_C(e) \) for all edges \( e \in Q \) induce the corresponding \( x \)-coordinates of all vertices of \( Q \).

So for each strip \( s_i \) (\( i = 0, \ldots, m \)) let \( y_i \) denote the \( y \)-coordinate of its lower boundary. For every edge \( e \in Q \) let \( t(e) \) and \( b(e) \) denote the index of the top- and bottommost strip, respectively, that is crossed by \( e \). Then \( \Delta_y(e) = y_{t(e)-1} - y_{b(e)} \). We propose the following linear program (LP) to solve
We assign to all vertices their corresponding $y$-coordinates from the solution of the LP. Then we compute for each edge $uv$ the correct $x$-coordinates of $u$ and $v$ from the vertical distance and the angle $\alpha_p(uv)$. This yields an embedding $\zeta$ that satisfies the constraints of Problem 2: (i) the $y$-order of all points is explicitly preserved in the LP, (ii) the $x$-coordinates are adjusted as to realize the given edge orientations, (iii) each edge is no shorter than its minimum length as this is modeled as a constraint in the LP, and (iv) the LP solution minimizes the stretching cost $c_s(\zeta)$.

Solving a linear program generally requires $O(n^{3.5}L)$ time [5], where $n$ is the number of variables and $L$ is the number of input bits; note that in our case we might have to deal with non rational numbers. Hence, the algorithm might be unsuitable.

### 3.4 Step 4: Concatenating Schematized Monotone Paths

At the end of step 3, we obtained for each subpath $(P_i, \pi_i)$ a $C$-oriented and orthogonal-order preserving embedding $\zeta_i$ with minimum schematization and stretching cost. In this step we concatenate the embedded paths $(P_i, \zeta_i)$ by adding up to three new path-link edges between any two adjacent subpaths $P_i$ and $P_{i+1}$ in order to obtain an embedding $\xi$ for a superpath $P'$ of $P$. The path-link edges are used to guarantee that $(P', \xi)$ is conflict-free, i.e., it has no self-intersections.

**Problem 3.** Given a sequence of $k$ embedded $x$- or $y$-monotone paths $(P_i, \zeta_i)$ with $1 \leq i \leq k$, find an embedding $\xi$ of $P' = P_1 \oplus \cdots \oplus P_k$, where $\oplus$ denotes the concatenation of paths, such that

(i) for each subpath $(P_i, \zeta_i)$, the embedding $\xi$ is a translation of $\zeta_i$ and

(ii) $(P', \xi)$ is a simple $C$-oriented path.

Our approach is based on iteratively embedding the subpaths $P_1, \ldots, P_k$. We ensure that in each iteration $i$ the embedding of $P_1 \oplus \cdots \oplus P_i$ remains conflict-free. For each $1 \leq i \leq k$ let $B_i$ denote the bounding box of $(P_i, \zeta_i)$. We show how to construct an embedding $\xi$ of $P'$ such that for any $i \neq j$ we have $B_i \cap B_j = \emptyset$. Consequently, since each individual $(P_i, \zeta_i)$ is conflict-free, $(P', \xi)$ is conflict-free as well. A key operation of the algorithm is shifting a subpath $P_i$ (or equivalently a bounding box $B_i$) by an offset $\Delta = (\Delta_x, \Delta_y) \in \mathbb{R}^2$. This is done by defining the lower left corner of each bounding box $B_i$ as its origin $o_i$ and storing the coordinates of $P_i$ relative to $o_i$, i.e., $\xi(v) = o_i + \zeta_i(v)$. Note that shifting preserves all local properties of $(P_i, \zeta_i)$, i.e., the orthogonal order as well as edge lengths and orientations.

Each iteration of our algorithm consists of two steps. First, we attach the subpath $P_i$ to its predecessor $P_{i-1}$. To that end, we initially place $(P_i, \zeta_i)$ such that the last vertex $u$ of $P_{i-1}$ and the first vertex $v$ of $P_i$ coincide. Then we add either two path-link edges (if the monotonicity directions of $P_{i-1}$ and $P_i$ are orthogonal) or three path-link edges (if $P_{i-1}$ runs in the opposite direction of $P_i$) between $u$ and $v$ and shift $B_i$ by finding appropriate lengths for the new edges such that $B_{i-1} \cap B_i = \emptyset$. Paths $P_{i-1}$ and $P_i$ are now conflict-free, but there may still exist conflicts between
These are resolved in a second step that, roughly speaking, "inflates" $B_i$ starting at $v$ until it has reached its original size. Any conflicting bounding boxes are "pushed" away from $B_i$ by stretching some of the path-link edges. In the following, we explain our procedures for attaching a subpath and resolving conflicts in more detail.

**Step 4a: Attaching a Subpath.** Without loss of generality, we restrict ourselves to the case that $P_{i-1}$ is an $x$-monotone path from left to right. Let $u$ be the last vertex of $P_{i-1}$ and $v$ be the first vertex of $P_i$. If $P_i$ is $y$-monotone we add a horizontal edge $e_1 = uu'$ with $\alpha_\xi(e_1) = 0^\circ$ connecting $u$ to a new vertex $u'$. Then we also add a vertical edge $e_2 = u'v$ with $\alpha_\xi(e_2) = 90^\circ$ if $P_i$ is upward directed and $\alpha_\xi(e_1) = 270^\circ$ if it is a downward path. Otherwise, if $P_i$ is $x$-monotone from right to left, we add two vertices $u'$ and $u''$ and three path-link edges $e_1 = uu'$, $e_2 = u'u''$, and $e_3 = u''v$ with $\alpha_\xi(e_1) = 0^\circ$, $\alpha_\xi(e_2) = 90^\circ$ if $P_i$ is above $P_{i-1}$ in $\pi$ or $\alpha_\xi(e_2) = 270^\circ$ otherwise, and $\alpha_\xi(e_3) = 180^\circ$. Note that technically we treat each path-link edge as having its own bounding box with zero width or height. It remains to set the lengths of the path-link edges such that $B_i \cap B_j = \emptyset$ by computing the vertical and horizontal overlap of $B_{i-1}$ and $B_i$. Figure 3 illustrates both situations.

**Step 4b: Resolving Conflicts.** After adding $P_i$ we have $B_{i-1} \cap B_i = \emptyset$. However, there may still exist conflicts with any $B_j$, $1 \leq j < i - 1$. In order to free up the space required to actually place $B_i$ without overlapping any other bounding box, we push away all conflicting boxes in three steps. For illustration, let $P_i$ be $x$-monotone from left to right, and let $v$ be the first vertex of $P_i$.

Each bounding box $B$ is defined by its lower left corner $ll(B) = (ll_x(B), ll_y(B))$ and its upper right corner $ur(B) = (ur_x(B), ur_y(B))$. In the first step we identify the leftmost box $B'$ (if any) that is intersected by a line segment that extends from $\xi(v)$ to the right with length equal to the width of $B_i$. For this box $B'$ we have $ll_y(B') \leq y_\xi(v) \leq ur_y(B')$ and $ll_x(B_i) \leq ll_x(B') \leq ur_x(B_i)$. If there is such a $B'$ let the offset be $\Delta_x = ur_x(B_i) - ll_x(B')$.

Now we shift all bounding boxes $B$ that lie completely to the right of $ll_x(B')$ to the right by $\Delta_x$. All horizontal path-link edges (which are also considered bounding boxes by themselves) that connect a shifted with a non-shifted path are stretched by $\Delta_x$ to keep the two paths connected. Note that there is always a horizontal path-link edge between any two subsequent paths.

Next, we inflate $B_i$, which is currently a horizontal line segment, downwards: we first determine the topmost conflicting box $B''$ (if any) below a horizontal line through $\xi(v)$, i.e., a box $B''$ whose $x$-range intersects the $x$-range of $B_i$ and for which $ll_y(B_i) \leq ur_y(B'') \leq y_\xi(v)$. If we find such a $B''$ we define the vertical offset $\Delta_{y1} = ur_y(B'') - ll_y(B_i)$. We shift all bounding boxes $B$ that lie completely below $ur_y(B'')$ downwards by $\Delta_{y1}$. All vertical path-link edges that connect a shifted
with a non-shifted box are stretched by $\Delta y_1$ in order to keep the two boxes connected. Again, there is always a vertical path-link edge between any two subsequent paths. Finally, we inflate $B_i$ upwards, which is analogous to the downward inflation. Figure 4 shows an example of Step 4b.

The approach explained above first inflates the box to the right, then downwards, and finally upwards. Of course, we could use any strategy of inflating the box. In fact, since different strategies yield different results, we could test a fixed number of strategies and keep the result that minimizes the overall increase in edge lengths.

In the following, we show that our algorithm indeed solves Problem 3.

**Lemma 2.** Step 4 computes a conflict-free embedding $\xi$ of $P' = P_1 \oplus \cdots \oplus P_k$.

**Proof.** We prove the theorem by induction. The first path $P_1$ has a conflict-free embedding due to Step 3 of our algorithm. Now, assume that $P_1 \oplus \cdots \oplus P_{i-1}$ is already embedded conflict-free. Step 4a attaches $P_i$ to $P_{i-1}$, such that $B_i \cap B_{i-1} = \emptyset$ holds. Thus, $P_i$ and $P_{i-1}$ do not have a conflict. Next, in Step 4b, the algorithm shifts all $B_j$ with $j < i$ such that $B_i \cap B_j = \emptyset$ in the end.

What remains to be shown is that our shifting-operation does not create new conflicts between any two boxes $B_j$ and $B_{j'}$ with $j, j' < i$. Consider, e.g., a shift to the right and assume that after the shift $B_j$ and $B_{j'}$ intersect, while they did not intersect before the shift. Clearly, if none of the boxes has been moved, they cannot intersect. So either the shift moved both boxes by the same offset $\Delta x$, or it moved only the rightmost of the two boxes to the right by $\Delta x$. There is no vertical movement. So in both cases the horizontal distance of the boxes does not decrease and it is impossible that $B_j$ and $B_{j'}$ intersect after the shift. The same observation holds for the vertical shifts. $\Box$

**Theorem 3.** Our algorithm computes a solution $(P', \xi)$ to Problem 3 by adding at most $3(k - 1)$ path-link edges to $P$. It can be implemented with a running time of $O(k^2 + n)$.

**Proof.** The correctness of the algorithm follows from Lemma 2 and due to the fact that the embeddings $(P_i, \xi_i)$ within the bounding boxes $B_i$ remain unchanged. Clearly, Step 4a adds at most three edges between any two subsequent paths, which shows the bound on the number of path-link edges.

![Fig. 4: Example for iteratively resolving conflicts induced by attaching $P_i$. First, we shift everything right of $L_x$ to the right by $\Delta x$. Then, we shift everything below $L_{y1}$ by $\Delta y_1$ downward, and finally, we shift everything above $L_{y2}$ upward.](image-url)
It remains to show the running time of $O(k^2 + n)$. For the position of each bounding box we maintain the coordinates of its lower left and upper right corners. In each iteration Step 4a requires constant time since it depends only on the overlap of the new bounding box and its immediate predecessor. Step 4b requires $O(k)$ time per iteration: When adding path $P_i$ we need to check for each box $B_j$ ($j < i - 1$) and the boxes of their respective path-link edges if and how they conflict with $B_i$. Once the required offset is computed, we shift $O(k)$ boxes by updating their lower left and upper right corners, as well as their origins. This is done in constant time per box and $O(k)$ time in total per iteration. So for all $k$ iterations the running time is $O(k^2)$. Finally, we obtain the embedding $\xi$ of $P'$ by computing the absolute coordinates of all vertices in $O(n)$ time.

Note that we can improve the running time of Step 4 if the number $I$ of occurring conflicts and shifts is small. More precisely, if $I < k^2 / \log^2 k$ we can use dynamic interval trees and dynamic binary search trees to store the segments and coordinates of all bounding boxes [3]. In this case, obtaining the bounding boxes that define the offsets for all shift operations can be done in $O(k \log^2 k + I)$ total time. Each stretch or shift of a box requires an update in these data structures, which can be done in $O(\log^2 k)$ time. Hence, for $I$ updates and $k$ queries we obtain a total running time of $O((I + k) \log^2 k + n)$ instead.

4 Conclusion

We presented a four step approach to schematize paths motivated from drawing route sketches: We split the path of length $n$ into $k$ monotone subpaths, embed each of them in a $C$-oriented way such that the orthogonal ordering is preserved. In a third step, we optimize edge lengths, followed by our final step, the concatenation of the subpaths such that the resulting embedding is intersection-free. Our approach has polynomial running time and is dominated by the optimization of the edge lengths using linear programming. Preliminary results from an implementation of our approach (see Fig. 1c) indicate that typically $n \leq 100$ for the road network of Germany, making our approach very efficient for drawing route sketches in practice. Moreover, our experiments also indicate that $k$ tends to be small, i.e., $k \leq 3$.

However, for usage in a real-world application, some problems remain. We need to label the edges, indicate turns, etc.; we plan to tackle all these problems. Moreover, it would be interesting to optimize the routes for simplicity in terms of driving and visualization rather than solely for travel time, i.e., a slightly longer route with less road changes might be preferable as it lowers the chance of making mistakes. Apart from schematizing a single route, it is another interesting problem to draw a whole set of alternative routes in a single sketch.

Acknowledgments. We thank David Eppstein, Bastian Katz, Maarten Löffler, Ignaz Rutter, and Markus Völker for discussions on the edge-length minimization problem.

References


