# Schematization of Networks

*Sergio Cabello*

*Mark de Berg*

*Marc van Kreveld*

institute of information and computing sciences, utrecht university
technical report UU-CS-2002-040
www.cs.uu.nl

# Schematization of Networks *

Sergio Cabello  Mark de Berg  Marc van Kreveld

Institute of Information and Computing Sciences, Utrecht University
P.O.Box 80.089
3508TB Utrecht
The Netherlands
{sergio,markdb,marc}@cs.uu.nl

**Abstract**

We study the problem of computing schematized versions of network maps, like railroad or highway maps. Every path of the schematized map has two or three links with restricted orientations, and the schematized map must be topologically equivalent to the input map. Our approach can handle several types of schematizations, and certain additional constraints can be added, such as a minimum vertical distance between two paths. Our algorithm runs in $O(n \log n)$ time, and experimental results showing the quality of the output are given.

## 1 Introduction

Automated cartography is an area of research that started in the sixties. Initially the topics consisted of automating certain tasks originally done by cartographers; later, the area mixed with the research on geographic information systems. One of the types of map that allows automated construction is the schematized map. Here, a set of nodes and their connections are displayed in a highly simplified form, since the precise shape of the connections and position of the points is not so important. To preserve the recognizability for map readers, the approximate layout must be kept, however.

This paper presents efficient algorithms for schematic map construction. The input we assume is a planar embedding of a graph consisting of polygonal paths between specified points called endpoints. It represents for instance a road or railroad network. The output consists of another planar embedding where all endpoints have the same positions, and every path is displayed as a two-link or three-link path where links are restricted to certain orientations (Figure 1). Furthermore, the output map should be equivalent to the input map in the sense that a continuous deformation exists such that no path passes over an endpoint during the transformation. This topological equivalence can also be expressed in the following way: our algorithm keeps the face structure in both embeddings. A consequence of this equivalence is that cyclic order of paths around endpoints is maintained. If a schematized map of the specified type doesn't exist, our algorithm reports this failure. The approach we describe works for several types of schematized paths. For example, we can use axis-parallel shortest paths with at most three links each, or allow links in the four main orientations (axis-parallel and angles of 45° and 135°). A vertical minimum separation distance between paths can be specified, and in certain cases we can allow or disallow that paths leaving the same endpoint partially share the first link. Details are given in Section 4. When the original map consists of $n$ segments, our algorithm runs in $O(n \log n)$ time.

The schematization problem has been studied before in several papers. Elroi [11, 12] describes an approach where the paths are first simplified, then they are placed on a grid to assure restricted orientations, and then crowded areas are locally enlarged to prevent too high density. No technicalities or running-time analysis are given in that paper. The paper by Neyer [16] describes a

*Supported by the Cornelis Lely Stichting.

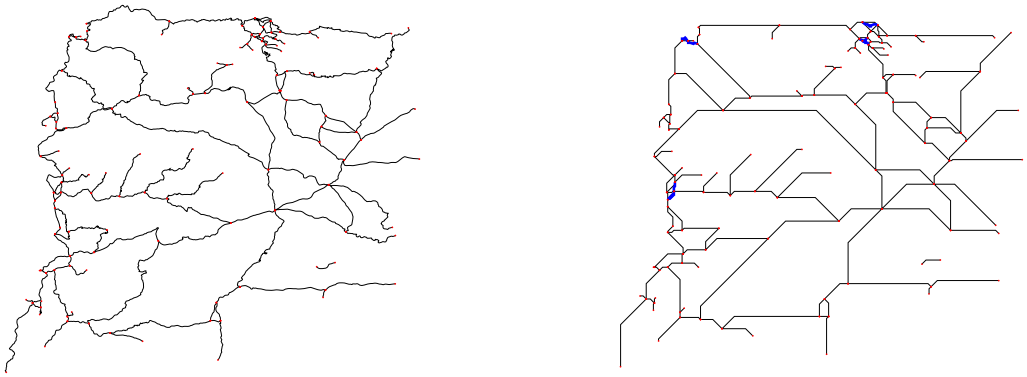

Figure 1: Northwest of Iberic Peninsula. Left: original map. Right: the schematized version made by the implementation.

line-simplification algorithm where the final paths must have links in one of $c$ given orientations only, and stay close enough to the original path. The algorithm minimizes the number $k$ of links in the output in $O(n^2 k \log n)$ time, and when it is applied to disjoint paths, the output may have intersections, which changes the topology of the map. Avelar and Müller [4] describe an iterative procedure that attempts to turn all links of an input network to one of the four main orientations. No upper bound on the running time can be given. Also, the output may contain links in other orientations than the desired ones. Barbowsky, Latecki and Richter [5] have also used an iterative discrete curve evolution, based on a local measure, to simplify the curves while keeping the local spatial ordering. Unlike the previous works, our approach gives a maximum number of links per path and is efficient. Unlike Neyer [16], we don't preserve proximity of paths to the input paths, but keep the positions of the endpoints and topological equivalence. Unlike Avelar and Müller [4], and Barbowsky, Latecki and Richter [5], we guarantee that only the four given orientations are used. A disadvantage is that our algorithm only gives a fully schematized map if one exists; otherwise, no output is given or certain paths are not schematized.

Other related research is map generalization for road networks and line simplification. However, the objectives in such problems are quite different. In general one doesn't consider achieving a given number of links per path and/or having restricted orientations. Also related is the research on VLSI layout design [13, 15], where the number of edges in the output are not considered, and research on graph drawing [9, 19], where the positions of the endpoints are usually not fixed. In a paper by Raghavan et al. [18], a wiring is made by connecting pairs of points by non-intersecting, 2-link orthogonal paths. In this case, only two different schematic paths are possible for each pair of points. The problem is solved in $O(n \log n + k)$ time, where $k$ is the number of potential conflicts. $k$ can be $\Theta(n^2)$ in the worst case. A recent, related topic that is relevant to schematization is the rendering of particular routes under queries [2, 3]. In this case, the paths are also simplified, but there is more flexibility to distort the input map because only the objects in the surroundings of the route are displayed.

Our paper is structured as follows. Section 2 describes the basic definitions and concepts. We explain in which sense the input map and output map are equivalent, by making use of homotopy type. Furthermore, we define an order between paths and we discuss its basic implications.

Section 3 explains how to compute the defined order among the paths of the input map. Globally, our algorithm works as follows. As it is done in [6] (see also [10]), we use an aboveness order among monotone pieces of the paths and we bring our problem into an orthogonal setting.

While keeping homotopy type and simplicity, we simplify each path to an $x$-monotone one. Then, we show that the order among paths in this map and in the original one are the same, and we use the map with $x$-monotone paths to compute this order.

Section 4 describes the method to place the schematized paths. We place each path following the computed order from top to bottom. Each path is placed at the topmost position that is still possible. This will leave the maximum freedom for later paths that need be placed. We can specify the type of schematized paths allowed and a vertical separation distance between paths, if desirable. If in the transformed map of Section 3 there is no order (a cycle is detected) among its paths, or the placement of some schematized path fails, then a schematized map of the desired type doesn't exist.

Section 5 shows some experimental results in order to evaluate the visual quality of the output provided by our algorithm (Figure 1). Finally, Section 6 gives a summary and directions for further research.

# 2 Equivalent maps: definition and basic properties

## 2.1 Equivalent paths and equivalent maps

A path is a continuous mapping $c : [0, 1] \to \mathbb{R}^2$, and the path is simple (no self-intersections) if the mapping is injective. For the rest of this paper we will restrict all paths to be polygonal paths. We use $|c|$ to denote the complexity of path $c$, defined as the number of edges it has.

**Definition 1** *A* (polygonal) map $M$ *is a set of simple polygonal paths* $\{c_1, \ldots, c_m\}$ *such that two paths do not intersect except at shared endpoints. A* monotone map *is a map where all paths are* $x$*-monotone.*

We use $P_M$ to denote the set of endpoints $\{c(0), c(1) \,|\, c \in M\}$, and $n = \sum_{i=1}^{m} |c_i|$ for the complexity of the whole map. To simplify presentation we assume that all vertices in the map have different $x$-coordinates; our algorithm can be adapted in a straightforward way for the general case.

Our goal is to construct schematized versions of a given map that are equivalent to it. Intuitively, two maps are equivalent if all paths can be transformed from one map to the other one in a continuous way, fixing the endpoints and without crossing any 'important point'. For example, if a road passes to the north of an important city, we don't want the schematized version to pass to the south of that city. Furthermore, we want the cyclic order of roads at crossings to stay the same. We let the important points for one path be the endpoints of the other paths; our algorithms can easily be adapted to take additional important points into account.

To formalize the approach and its properties we need the topological concept of equivalent (or homotopic with fixed endpoints) paths [8, 14].

**Definition 2** *Given a set of points $P$, two paths $c$ and $\tilde{c}$ with the same endpoints are equivalent in $\mathbb{R}^2 \setminus P$ if and only if there exists a continuous function $F : [0, 1] \times [0, 1] \to \mathbb{R}^2 \setminus P$ such that:*

- *$F(0, t) = c(t)$ and $F(1, t) = \tilde{c}(t)$, for all $t \in [0, 1]$ (the first path is $c$, the final path is $\tilde{c}$),*
- *$F(s, 0) = c(0) = \tilde{c}(0)$ and $F(s, 1) = c(1) = \tilde{c}(1)$, for all $s \in [0, 1]$ (the endpoints are fixed).*

*Two maps $M = \{c_1, \ldots, c_m\}$ and $\tilde{M} = \{\tilde{c}_1, \ldots, \tilde{c}_m\}$ are* equivalent maps *if and only if for some renumbering of the paths in $\tilde{M}$, paths $c_i$ and $\tilde{c}_i$ are equivalent in $(\mathbb{R}^2 \setminus P_M) \cup \{c_i(0), c_i(1)\}$ for all paths $c_i \in M$.*

The problem of schematizing a map can now be restated as follows: given a map, compute an equivalent map whose paths are of a certain type (such as axis-aligned, $x$-monotone, 3-links, etc) and have certain properties (such as a minimum vertical distance between two schematized paths).

## 2.2 Order among paths

For a point $p$, we use $l_p^+$ and $l_p^-$ to denote the vertical halfline with point $p$ as lowest and highest point respectively. We next define aboveness of paths, which is an invariant among equivalent paths and equivalent maps.

**Definition 3** *Let $c$ be a path. A point $p \in \mathbb{R}^2$ with $p \notin c$ is* above (below) *$c$ if for every equivalent path $c'$ in $\mathbb{R}^2 \setminus \{p\}$ the intersection of $l_p^-$ (respectively $l_p^+$) and $c'$ is nonempty. A path $c_i$ is* above *$c_j$ if $c_i(0)$ or $c_i(1)$ is above $c_j$, or if $c_j(0)$ or $c_j(1)$ is below $c_i$.*

We would like to remark that, in this definition, to decide whether a point is above or below a path, we do not take into account any other point, that is, our universe is reduced exclusively to the point and the path. Observe that in some cases, it is possible that a point is both above and below a curve. In other cases, no order is present. See Figure 2 for an example.

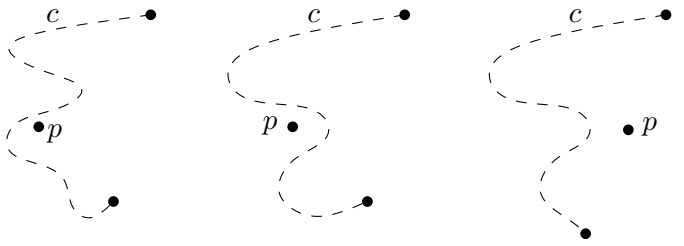


Figure 2: Example of relations between points and paths. Left: $p$ is both above and below $c$. Center: $p$ and $c$ have no relation. Right: $p$ is below $c$.

**Lemma 1** *The above-below relation among paths is an invariant between equivalent maps.*

**Proof:** The equivalence of paths is an equivalence relation [8, 14]. For this reason, for any point $p \in \mathbb{R}^2$, equivalent paths in $\mathbb{R}^2 \setminus \{p\}$ have the same relation with respect to the point $p$.

Let $M$ and $\tilde{M}$ be equivalent maps. For any path $c_i \in M$, let $\tilde{c}_i \in \tilde{M}$ be its corresponding path. Because $c_i$ and $\tilde{c}_i$ are equivalent in $(\mathbb{R}^2 \setminus P_M) \cup \{c_i(0), c_i(1)\}$, we have that for any endpoint $p \in P_M \setminus \{c_i(0), c_i(1)\}$ they are also equivalent in $\mathbb{R}^2 \setminus \{p\}$. So, any endpoint different from $c_i(0), c_i(1)$ has the same above-below relation with $c_i$ and $\tilde{c}_i$, and this implies the result. □

## 2.3 Canonical sequence

From an algorithmic point of view, it is not clear how Definition 2 can be used to test if two paths $c$ and $\tilde{c}$ are equivalent in $\mathbb{R}^2 \setminus P$. It is also not clear how Definition 3 decides if a point $p \in P$ has some above-below relation with path $c$. Canonical sequences [6] appear to be useful for these problems.

Let $P$ be a set of points, and consider the rays $l_p^+, l_p^-$ for all points $p \in P$. For a given path $c$, start walking from $c(0)$ and, while following $c$, write the sequence of rays that are crossed. If in this sequence, we repeatedly apply the operation of removing two adjacent rays that are identical, we get what is called the *canonical sequence* of $c$ with respect to $P$. See Figure 3, left, for an example. Using the concept of *universal cover* [8, 14], it is not difficult to prove the following result (the proof is given in [6]).

**Lemma 2** *Two paths with the same endpoints have the same canonical sequence with respect to $P$ if and only if they are equivalent in $\mathbb{R}^2 \setminus P$.*

In the particular case when $P$ consists of only one point $p$, any path $c'$ equivalent to $c$ in $\mathbb{R}^2 \setminus \{p\}$ has to cross the rays $l_p^+$ or $l_p^-$ that appear in the canonical sequence of $c$ with respect to $\{p\}$. Thus, a point $p$ is below (above) a path $c$ if and only if $l_p^+$ (respectively $l_p^-$) appears in the canonical sequence of $c$ with respect to $\{p\}$. See Figure 3, right, for an example.

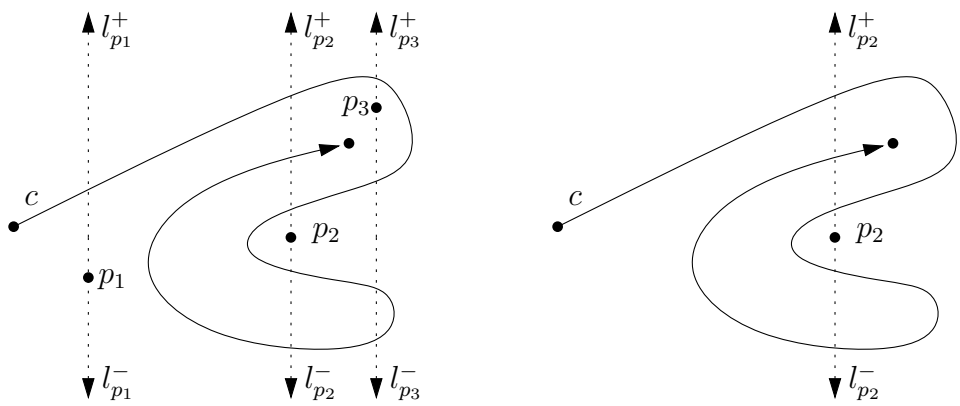


Figure 3: Left: example of a canonical sequence. The original sequence is $l^+_{p_1} l^+_{p_2} l^+_{p_3} l^-_{p_3} l^+_{p_2} l^-_{p_2} l^-_{p_3} l^-_{p_3} l^-_{p_2} l^+_{p_2}$ and thus its canonical one is $l^+_{p_1} l^+_{p_2} l^+_{p_3} l^-_{p_3}$. Right: example comparing $p_2$ with $c$. The sequence to consider is $l^+_{p_2} l^+_{p_2} l^-_{p_2} l^-_{p_2} l^+_{p_2} \equiv l^+_{p_2}$, and we can conclude that $p_2$ is below path $c$.

## 2.4 Above-below relations in monotone maps

In [6, 17] the following aboveness order is used between $x$-monotone paths $a$ and $b$: $a \succ b$ if and only if there are points $(x, y_a) \in a$ and $(x, y_b) \in b$ with $y_a > y_b$. It is important to note that this relation is the same as the one given in Definition 3 for the case of disjoint, $x$-monotone paths. From [6, 17] the next result follows.

**Lemma 3** *For a monotone map $M$, the above-below relation among paths is acyclic. Furthermore, if $M$ has complexity $n$, a total order extending this relation can be computed in $O(n \log n)$ time.*

**Corollary 1** *For a given map $M$, if there is no partial order among its paths, then no monotone map can be equivalent to $M$.*

**Proof:** This follows from Lemma 1 and Lemma 3. □

Because we are only interested in schematic maps with $x$-monotone paths, this corollary implies that the algorithm to be developed may report the impossibility of the schematic map. For monotone maps, the above-below relation provides a complete characterization up to equivalence, which is shown in the next lemma.

**Lemma 4** *Let $M = \{c_1, \ldots, c_m\}$ and $\tilde{M} = \{\tilde{c}_1, \ldots, \tilde{c}_m\}$ be two monotone maps such that the paths $c_i$ and $\tilde{c}_i$ have the same endpoints. Then, the maps $M$ and $\tilde{M}$ are equivalent if and only if they define the same above-below relation.*

**Proof:** One implication is provided by Lemma 1. For the other implication, we show that if $M$ and $\tilde{M}$ are not equivalent, then they define a different above-below relation. Observe that if $c_i$ and $\tilde{c}_i$ are not equivalent in $(\mathbb{R}^2 \setminus P_M) \cup \{c_i(0), c_i(1)\}$, it is due to the existence of some point $p \in P_M \setminus \{c_i(0), c_i(1)\}$ in the regions between $c_i$ and $\tilde{c}_i$ (these regions are well-defined because both $c_i$ and $\tilde{c}_i$ are $x$-monotone, see Figure 4 left). It follows that $p$ has a different above-below relation with $c_i$ and $\tilde{c}_i$, and for any path $c_j$ with $p$ as an endpoint, the relation between paths $c_j$ and $c_i$ will be different from the relation between $\tilde{c}_j$ and $\tilde{c}_i$. □

Observe that this result is not true if we remove the monotonicity conditions, as shown in Figure 4 right.

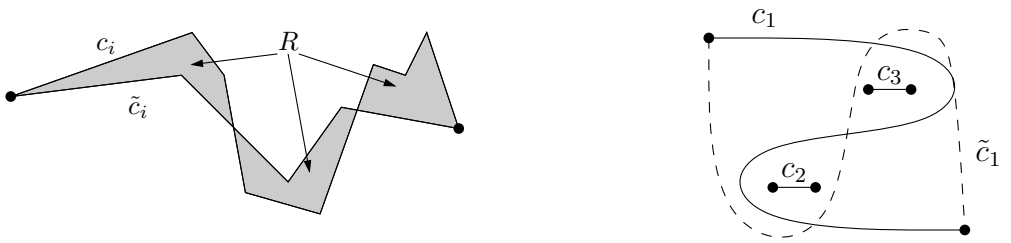


Figure 4: Left: two $x$ monotone paths $c_i, \tilde{c}_i$ that share the same endpoints have a well-defined region $R$ between them (gray in figure). If there is a point $p$ inside $R$, then $c_i$ and $\tilde{c}_i$ cannot be equivalent in $\mathbb{R}^2 \setminus \{p\}$. Right: maps $\{c_1, c_2, c_3\}$ and $\{\tilde{c}_1, c_2, c_3\}$ define the same above-below relations but they are not equivalent.

# 3 Computing order in a map

The straightforward approach of comparing each pair of paths to decide their relation gives a worst-case quadratic time algorithm. In this section we show how to compute a total order extending the partial order defined in the previous section in $O(n \log n)$ time. Since sorting real numbers can be reduced to compute the order in a map, our approach is asymptotically optimal. Some of the basic ideas explained here are taken from [6]. Firstly we convert the path into what is called a rectified map. Secondly, we transform the rectified map into one with $x$-monotone pieces, and finally we compute the order in this map.

## 3.1 Rectified maps

**Definition 4** *A set of paths $M' = \{c'_1, \ldots, c'_m\}$ is a* rectification *of a map $M = \{c_1, \ldots, c_m\}$ if:*

- *$M'$ is a map (its paths only intersect in common endpoints);*
- *the complexity of map $M'$ is linear in the complexity of map $M$;*
- *paths $c'_i$ are made of axis-aligned segments;*
- *paths $c_i$ and $c_j$ have the same above-below relation as $c'_i$ and $c'_j$.*

A map $M = \{c_1, \ldots, c_m\}$ can be rectified in the following way. Decompose each path $c_i \in M$ into monotone pieces $c_i^1, \ldots, c_i^{k_i}$. By promoting every locally leftmost or rightmost vertex of a path to an endpoint, we make a monotone map $M_{mono}$ with the set of pieces $\{c_1^1, \ldots, c_1^{k_1}, \ldots, c_m^1, \ldots, c_m^{k_m}\}$ as the new set of paths. Because $M_{mono}$ has complexity $O(n)$, we can compute—among the pieces $c_i^j$—a total order extending the partial order in $O(n \log n)$ time (Lemma 3).

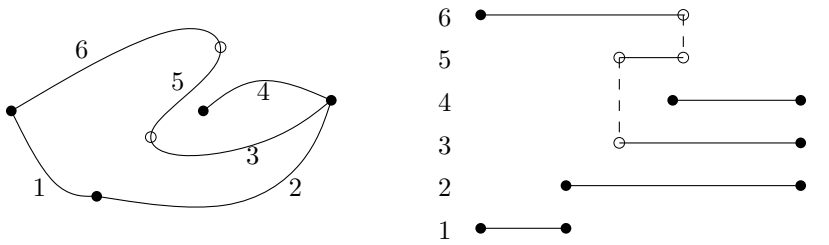


Figure 5: A map with its paths partitioned into monotone pieces, and their ranks. Right, the corresponding rectified map.

Using the rank of each piece in the total order, we can construct the rectified version of a path, see Figure 5. Let $c_i^j$ be a piece with left endpoint $(p_x, p_y)$, right endpoint $(q_x, q_y)$, and rank $r$. Then we make the horizontal segment $h_i^j = [p_x, q_x] \times r$ in the rectified map. We form the path $c_i'$ that joins $h_i^1, \ldots, h_i^{k_i}$ by connecting the endpoints of every two consecutive horizontal segments by a vertical segment. We denote by $M'$ the collection of all such paths, that is $M' := \{c_1', \ldots, c_m'\}$. Observe that an endpoint sharing several paths will be mapped in several points vertically above each other in $M'$.

**Lemma 5** *Given a map $M$ of complexity $n$, we can construct a rectification $M'$ in $O(n \log n)$ time.*

**Proof:** The computation of $M'$ as described takes $O(n \log n)$ time if we use Lemma 3. It is also clear from the construction that $M'$ satisfies the first three conditions to be a rectification of $M$. It remains to show that it also fullfills the fourth condition.

Consider the canonical sequence $s$ of the path $c_i$ with respect to the endpoint $c_j(0)$ and the canonical sequence $s'$ of the path $c_i'$ with respect to $c_j'(0)$. From the construction of $M'$ it follows that replacing in the sequence $s$ each occurrence of $l^+_{c_j(0)}$ (respectively $l^-_{c_j(0)}$) by $l^+_{c_j'(0)}$ (respectively $l^-_{c_j'0)}$) we get the sequence $s'$. We can conclude by Lemma 2 that the above-below relation of $c_j(0)$ to $c_i$ is identical to the above-below relation of $c_j'(0)$ to $c_i'$. The same argument applies to the relation of $c_j(1)$ to $c_i$ and the relation of $c_i(0), c_i(1)$ to $c_j$, and thus the above-below relation of $c_i$ and $c_j$ is the same as that of $c_i'$ and $c_j'$. □

## 3.2 Computing order using a rectified map

Let $M' = \{c_1', \ldots, c_m'\}$ be a map such that all segments are axis-aligned. In [6] an algorithm rcp (shorthand for rectified_canonical_path) is presented that transforms the path $c_i'$ into another path $\mathsf{rcp}(c_i')$ with the following properties:

1. $\mathsf{rcp}(c_i')$ is equivalent to $c_i'$ in $(\mathbb{R}^2 \setminus P_{M'}) \cup \{c_i'(0), c_i'(1)\}$;
2. $|\mathsf{rcp}(c_i')| \leq |c_i'|$;
3. if $c_i'$ and $c_j'$ don't intersect, then $\mathsf{rcp}(c_i')$ and $\mathsf{rcp}(c_j')$ do not intersect either;
4. $\mathsf{rcp}(c_i')$ has the minimum possible number of $x$-monotone pieces that any path equivalent to $c_i'$ in $(\mathbb{R}^2 \setminus P_{M'}) \cup \{c_i'(0), c_i'(1)\}$ can have.

To use rcp, we first have to preprocess the endpoints $P_{M'}$ of the map $M'$ for the following type of range queries: given a query rectangle $R$, find the leftmost (or rightmost) point in $P_{M'}$ that lies in $R$. This can be done in $O(|P_{M'}| \log |P_{M'}|)$ time [7], and then, it takes $O(|c_i'| \log |P_{M'}|)$ time to compute $\mathsf{rcp}(c_i')$ [6]. Therefore, to compute $\mathsf{rcp}(c_i')$ for all paths $c_i' \in M'$ takes

$$O(|P_{M'}| \log |P_{M'}|) + \sum_{c_i' \in M'} O(|c_i'| \log |P_{M'}|) = O(|M'| \log |P_{M'}|) = O(|M'| \log |M'|).$$

This is the basic ingredient for the main result of this section.

**Theorem 1** *For a map $M$ of total complexity $n$, we can decide in $O(n \log n)$ time whether an equivalent, monotone map exists.*

**Proof:** Given the map $M = \{c_1, \ldots, c_m\}$, we start by making a rectification of it, $M' = \{c_1', \ldots, c_m'\}$, and then we use rcp to compute $N := \{\mathsf{rcp}(c_1'), \ldots, \mathsf{rcp}(c_m')\}$. Observe that by Definition 4, Lemma 5, and the first three properties of rcp, it follows that $N$ is a map with complexity $O(n)$, it can be constructed in $O(n \log n)$ time, and it has the same above-below relations among its paths as $M$ does. We claim that $N$ is monotone if and only if $M$ admits an equivalent, monotone map.

Recall that a point $p \in P_M$ that is an endpoint of several paths in $M$ is mapped into several endpoints in $M'$. Let $p'$ be this set of endpoints. By construction, no segment of $M'$ can pass between two points of $p'$, and when considering the canonical sequences of a path $c'_i$ with respect to $P_{M'} \setminus \{c'_i(0), c'_i(1)\}$ we can just consider all points in $p'$ as one point, which we also denote by $p'$. With this notation, if we replace in the canonical sequence of $c_i$ with respect to $P_M \setminus \{c_i(0), c_i(1)\}$ each occurrence of $l_p^+$ by $l_{p'}^+$ and $l_p^-$ by $l_{p'}^-$, where $p$ is any point in $P_M \setminus \{c_i(0), c_i(1)\}$, then we get the canonical sequence of $c'_i$ with respect to $P_{M'} \setminus \{c'_i(0), c'_i(1)\}$.

If $M$ admits an equivalent, monotone map, then for each path $c_i \in M$, the canonical sequence of $c_i$ with respect to $P_M \setminus \{c_i(0), c_i(1)\}$ doesn't contain two rays emanating from the same point. But this is equivalent to saying that the canonical sequence of $c'_i$ with respect to $P_{M'} \setminus \{c'_i(0), c'_i(1)\}$ doesn't contain two rays emanating from the same point, which, by property 4 of $\mathsf{rcp}$, implies that $\mathsf{rcp}(c'_i)$ is $x$-monotone.

Conversely, if $N$ is a monotone map, then for each path $c_i \in M$ there is a path $\tilde{c}_i$ that is $x$-monotone and equivalent to $c_i$ in $(\mathbb{R}^2 \setminus P_M) \cup \{c_i(0), c_i(1)\}$. Two paths $\tilde{c}_i$ and $\tilde{c}_j$ can intersect, but we will show how to remove these intersections in order to get a map. Observe that when two paths intersect, they do so at least twice (tangencies and common endpoints do not count as intersections): if $\tilde{c}_i$ and $\tilde{c}_j$ would intersect only once, then $\tilde{c}_i$ would be both above and below $\tilde{c}_j$ (see Figure 6 left), but then also $\mathsf{rcp}(c'_i)$ would be above and below $\mathsf{rcp}(c'_j)$, which is not possible because $N$ is a monotone map by hypothesis. On the other hand, no point $p \in P_M$ can lie in the region that $\tilde{c}_i, \tilde{c}_j$ define between their first and second intersection (see Figure 6 right), otherwise $\tilde{c}_i$ would be both above and below $\tilde{c}_j$. Therefore, we can deform $\tilde{c}_i$ and $\tilde{c}_j$ into other paths (we also use $\tilde{c}_i, \tilde{c}_j$ for the new paths) and we reduce the total number of intersections by two (see Figure 6 right). The new path $\tilde{c}_i$ is also equivalent to $c_i$ in $(\mathbb{R}^2 \setminus P_M) \cup \{c_i(0), c_i(1)\}$, and the similar statement holds for $\tilde{c}_j$. Each time we apply this process we reduce the number of intersections in $\{\tilde{c}_1, \ldots \tilde{c}_m\}$, and thus if we repeat this process for each pair of paths that intersect, we will eventually end up with a set of $x$-monotone paths that do not have intersections (except at shared endpoints). This is a monotone map that is equivalent to the original one. □

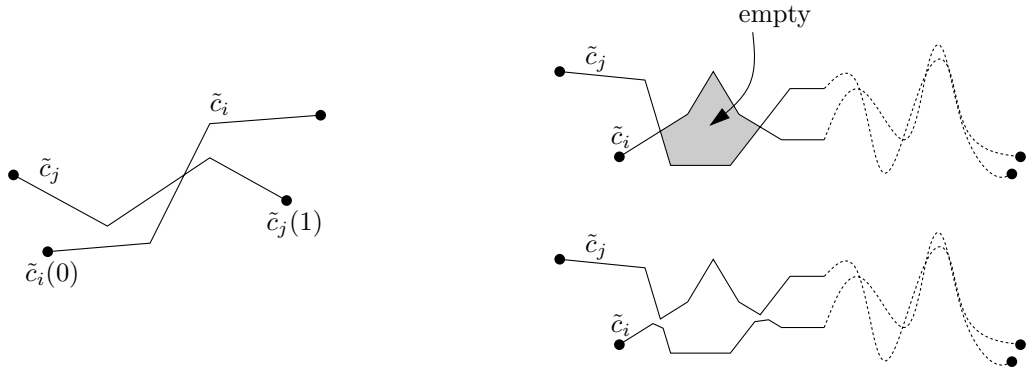


Figure 6: Left: if the paths $\tilde{c}_i$ and $\tilde{c}_j$ only intersect once, then they have a cyclic above-below relation. In this example, because the endpoint $\tilde{c}_i(0)$ is below $\tilde{c}_j$, the path $\tilde{c}_i$ is below $\tilde{c}_i$, but also $\tilde{c}_j(1)$ is below $c_i$, and so $\tilde{c}_j$ is below $\tilde{c}_i$. Right: if $\tilde{c}_i$ and $\tilde{c}_j$ intersect more than twice, the region between the first two intersections is well-defined (grey in the picture). This region cannot contain any point $p \in P_M$, otherwise $\tilde{c}_i$ is above $p$ which is above $\tilde{c}_j$, and we get a cyclic order. Then we can remove the intersection continuously in $(\mathbb{R}^2 \setminus P_M) \cup \{c_i(0), c_i(1)\}$.

**Corollary 2** *If for a map $M$ there is an equivalent, monotone map, then we can compute a total order extending the partial order among its paths in $O(n \log n)$ time.*

**Proof:** The order among paths in map $M = \{c_1, \ldots, c_m\}$ is the same as the order in map $N = \{\mathsf{rcp}(c'_1), \ldots, \mathsf{rcp}(c'_m)\}$. The hypothesis implies that $N$ is monotone, and by Lemma 3 we can therefore compute such a total order. □

# 4 Placing paths in the schematic map

In the previous sections we determined a partial top-to-bottom order on the paths in the input map. This section concentrates on the actual placement of the paths: we show that placing the paths one by one, in the computed order, where each path is placed topmost (that is, as high as possible) will result in a schematic map if one exists. We use the notation $X_1X_2X_3$-path ($X_1X_2$-path), with $X_i \in \{\text{H, V, D}\}$ to denote a 3-link (respectively 2-link) path whose $i$-th link is of type $X_i$. Here type H means horizontal, type V means vertical, and type D means diagonal.

To explain the main features of the placement we will describe the algorithm for one concrete case: 3-link {HDH, VDV}-paths that are $L_2$-shortest. This type of paths are shown in Figure 7 left and center, and a map using this type of paths is in Figure 8 (a). Later we will generalize to other types of paths. The idea is to incrementally place the paths from top to bottom respecting the order, and maintain the lower envelope of the previously placed paths in a binary search tree $\mathcal{T}$. The tree stores in its leaves, ordered from left to right, the vertices and segments of the lower envelope by increasing $x$-coordinate. When adding a path, we search with the left and right endpoints in $\mathcal{T}$, ending in two leaves $\mu$ and $\mu'$. We collect the part of the lower envelope in between. The new path must be below the collected pieces of the lower envelope. If one of the endpoints of the new path is above the lower envelope stored at $\mu$ or $\mu'$, the algorithm fails and no schematization exists. Otherwise, we can determine the topmost placement of the new path, with or without a minimum vertical separation distance. By topmost placement we mean the placement that makes the new lower envelope as high as possible. Again we may fail to find such a path if a previously placed path will be intersected by the new path, or the desired separation distance cannot be attained. If the new path can be placed, it will replace the pieces of the lower envelope we collected, except, possibly, for the outermost two. These may be truncated horizontally. In the tree $\mathcal{T}$, this comes down to deleting the leaves in between $\mu$ and $\mu'$. Up to three new leafs are inserted instead. Appropriate updates have to be done at $\mu$ and $\mu'$.

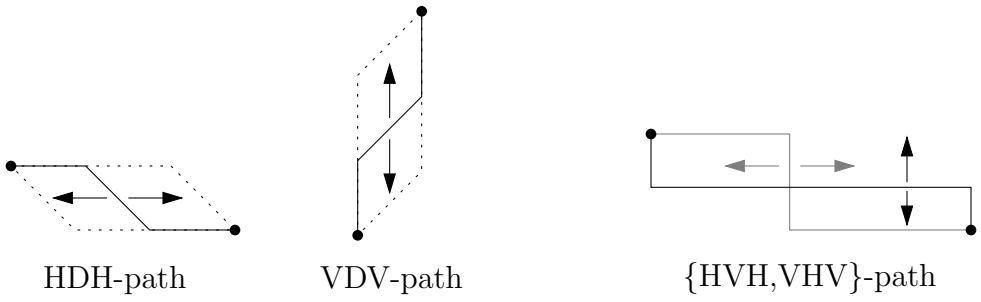


Figure 7: Some types of schematic paths.

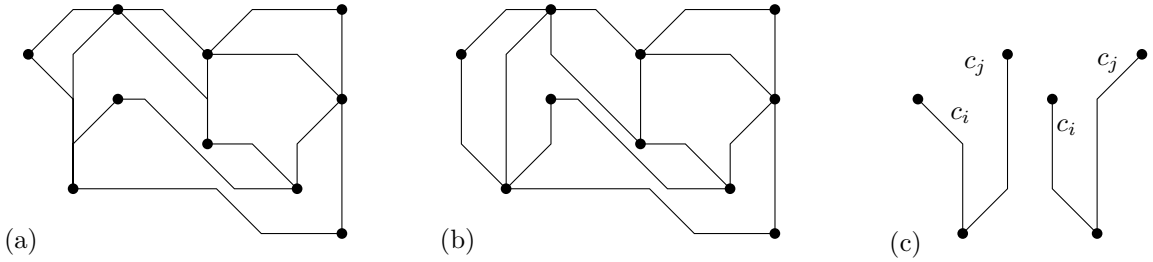


Figure 8: (a) A schematized map with 3-link, {HDH, VDV}-paths with shared departure from common endpoints; topmost placement. (b) Same but without shared departure. (c) The indecision in aboveness for $c_i$ and $c_j$ when shared departure is not allowed.

In this concrete case with {HDH, VDV}-paths that are $L_2$-shortest, the endpoints of the next path to be placed provides us a decision between a HDH-path or a VDV-path, because the path has to be $L_2$-shortest: we need a HDH-path when the vertical distance between the endpoints is smaller than the horizontal distance, and otherwise we need a VDV-path. In general, this property doesn't hold for certain combinations of types of schematic paths, for example {HVH, VHV}-paths that are $L_2$-shortest (Figure 7). In this case, whether we place a HVH-path or a VHV-path can influence whether we can place another, later schematic path or not. To understand what types of schematic paths our method can handle, we make the following definitions.

**Definition 5** *A* schematic map model *specifies, for any two points $p, q$ in the plane, the collection of (schematic) paths with $p$ and $q$ as endpoints that can be used. A schematic map model is* $x$-monotone *if all paths in any collection that it specifies are $x$-monotone paths. A schematic map model is* ordered *if, for any two points $p, q$, and any two paths $c_1, c_2$ in the collection specified by the model for $p, q$, no two points $a, b$ exist such that $a$ is above $c_1$ and below $c_2$, and $b$ is below $c_1$ and above $c_2$.*

In Figure 7, one can see that a model that allows shortest {HVH, VHV}-paths is not ordered: points $a$ and $b$ can be placed in the two distinct bounded regions between the paths. However, $L_2$-shortest {HDH, VDV}-paths result in an ordered $x$-monotone schematic map model. Our algorithm can handle any ordered $x$-monotone schematic map model. Examples are {VDH, HDV, VDV, HDH}-paths that are $L_1$-shortest, {VH,HV}-paths, and {HVH, VHV}-paths where the distance between the endpoints determines which of the two types is used (for instance, when the horizontal distance between the endpoints is larger than the vertical distance, use a HVH-path, otherwise a VHV-path). Links may degenerate to fewer links by using zero-length segments.

Besides the types of schematic paths, we may also specify a minimum vertical separation distance between two paths that don't have any shared endpoint. Observe that horizontal distance cannot be taken into account because this doesn't relate to the ordering used among paths.

In most cases we can specify that two schematic paths with the same endpoint (shared endpoint) may not depart in the same direction from that endpoint. However, in some cases it makes a difference which schematized path is placed first, see Figure 8 (c). In a degenerate sense, the paths are above and below each other, giving a cycle in the placement order. Therefore, our algorithm cannot forbid shared departure in the vertical upward direction in all cases. Disallowing shared departure in the non-vertical directions can be incorporated efficiently, however, because the above-below relations give the placement order. Shared departure in the vertical downward direction is automatically avoided if this possible, because every schematic path is placed topmost.

**Theorem 2** *Given a map $M$ with complexity $n$, and an ordered $x$-monotone schematic map model, a minimum vertical separation distance specified, and optionally, shared horizontal departure disallowed, we can compute in $O(n \log n)$ time a schematized map equivalent to $M$ satisfying the separation distance and shared departure conditions as specified, or report that no such map exists.*

**Proof:** By Theorem 1, we can detect in $O(n \log n)$ time if $M$ has an equivalent, monotone map. In the negative case, no schematization of $M$ is possible in an $x$-monotone schematic map model and we are done. Otherwise we can compute a total order extending the partial order among the paths in $M$ in $O(n \log n)$ time by Corollary 2. By Lemma 4 we have that any monotone map with this order among its paths is equivalent to $M$. In particular, a map constructed with top-to-bottom placement as detailed in this section is equivalent to $M$.

It remains to show that we can place top-to-bottom all the schematic paths in $O(n \log n)$ time. Recall that we use a binary search tree $\mathcal{T}$ to represent the lower envelope of the placed schematic paths. To insert a new path, we find the two leaves $\mu$ and $\mu'$ with the $x$-coordinates of the endpoints of the new path in $O(\log n)$ time. Assume that $k$ leaves lie in between $\mu$ and $\mu'$ in $\mathcal{T}$. Then we can determine in $O(k)$ time whether a placement of the new path exists, in any ordered $x$-monotone schematic map model, and with the separation distance and shared departure conditions. If a placement meeting the conditions doesn't exist, we can stop and report failure.

Otherwise, we determine the topmost placement within the same time bound, asymptotically. The updating of $\mathcal{T}$ involves the deletion of $k$ leaves, the insertion of $O(1)$ leaves, and the updating of two leaves $\mu$ and $\mu'$. This takes $O((k+1)\log n)$ time. Since in total $O(n)$ leaves are inserted, and any leaf can be deleted only once, it follows that the total running time of placing the schematic paths is $O(n \log n)$.

If at some moment the algorithm cannot place the next schematic path because we would violate some condition, like placement closer than the specified minimum separation distance, or the placement would introduce an intersection, then the schematic map as desired doesn't exist. This follows from the fact that we maintain the topmost lower envelope among all possible placements of the previously placed schematic paths, which is well-defined in an ordered $x$-monotone schematic map model. □

We always need to ensure that our original map admits an equivalent, monotone map. Otherwise, if we just compute the above-below relation among paths and then we place the schematic paths in the way that will be explained in this section, we can construct a schematic map that is not equivalent to the original one (see Figure 4 right).

# 5 Experimental results

In order to test the quality of the output when applying the aforementioned ideas, a prototype has been implemented as a Java applet and can be seen on the web [1]. Although some of the techniques to speed up the algorithm that were explained are not used, the time of computation doesn't appear to be problematic.

Some editing tools were included in the prototype in order to make it more flexible and interactive for the user. Recall that the algorithm described above doesn't move the endpoints of paths, but some adjustments may be necessary. For example, it appeared to be useful to identify and merge the endpoints of paths when they are very close. Observe that our algorithm would report failure in the cases that the paths have a slight overlap in the vertical direction, or they are closer in the vertical direction than allowed, see Figure 9. When some path cannot be schematized, the prototype draws the original path. Also, the presence of intersections close to endpoints plays an important role, because this prevents an ordering of the paths. Sometimes, this type of problem, which can be present in inaccurate data, can be resolved by editing, see Figure 9.

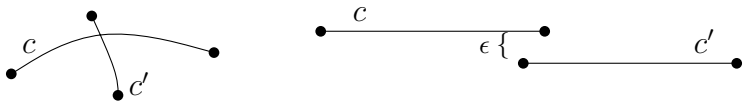


Figure 9: Some editing of the original map is necessary if it is topologically not correct or it has inaccuracies. Left: there is no order among paths. Right: the minimum separation distance between two paths cannot be respected.

From an aesthetic point of view, several practical improvements can be done during the placing of paths. For example, given the minimum vertical distance $d$ to be kept between the schematic paths, this applet attempts to find the maximum distance $d' \geq d$ that permits the schematization of as many paths as $d$ does. This improvement is done by a binary search over the integers (distance in pixels) to find this distance $d'$. Other improvements are to place paths in top-to-bottom and bottom-to-top order simultaneously, where the bottommost paths are schematized in the lowest placement possible. This adaptation yields maps that are more "rounded" than when all schematized paths are placed topmost. There are many other improvements that could be implemented too, like replacing corners by circular arcs to get $C^1$ continuity, parallel departure from endpoints represented by rectangles or bars, and so on.

The prototype was tested on maps of Canada, Ireland and Spain. They consist of between 769 and 1612 segments, and the time of computation is less than 3 seconds on a standard PC. The

most time consuming part is the binary search to find the optimal distance, as explained at the end of previous paragraph. In all examples and results in this section, the smallest allowed vertical distance between two schematized paths was set to 10 pixels. Some of the results can be seen in Figures 10–12, where paths that could not be schematized are shown in their original shape.

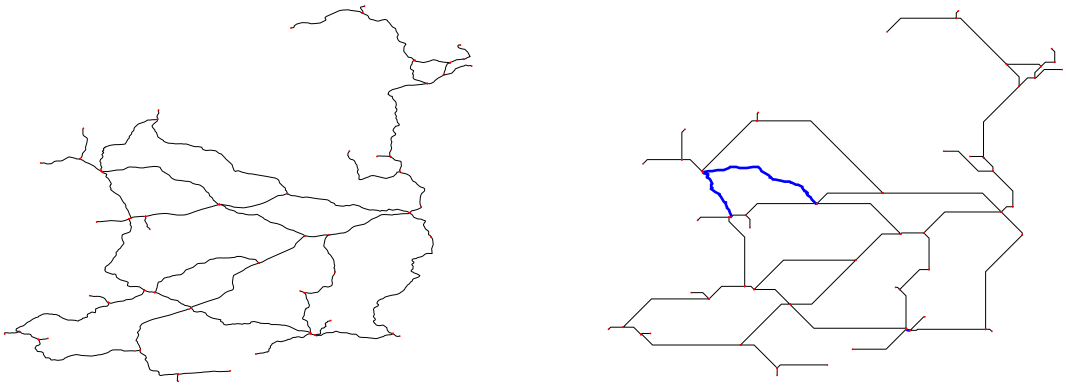

Figure 10: Ireland. Left: original railway map. Right: schematic map with {HDH, VDV}-paths and without shared departure.

For the maps of Ireland and Canada, a top-to-bottom placement order exists and hence, could be computed without any editing. But the map of Spain doesn't have a placement order. This is due to inaccuracies in the data, which is topologically not correct. Different paths had intersections that were not endpoints. After a few modifications with the editing tool, it was possible to compute a placement order.

As can be seen in the pictures, some schematic paths cannot be placed because they would intersect some other one already placed, give shared departure, or would not respect the specified distance. It appears in the figures that more schematic paths could be placed, like the two non-schematized paths in Figure 10. However, there are two endpoints very close together at the upper end of the non-schematized paths, which would yield a placement too close to another path with different endpoints. Editing of the original map would be necessary to resolve this. In Table 1 we can see how many paths were present originally, and which percentage of these couldn't be placed. After analyzing the pictures in the prototype, which allows us to zoom in the original map, one can see that the biggest source of problems in all maps is indeed that some endpoints are very close together.

| | Total number of paths | Schematic paths when shared departure is not allowed | Schematic paths when shared departure is allowed |
|---|---|---|---|
| Figure 10 right | 65 | 61 | 61 |
| Figure 11 right | 100 | 76 | 79 |
| Figure 12 center | 155 | 115 | 134 |
| Figure 12 bottom | 155 | 139 | 139 |

Table 1: Number of paths that were schematized for the maps shown in the figures.

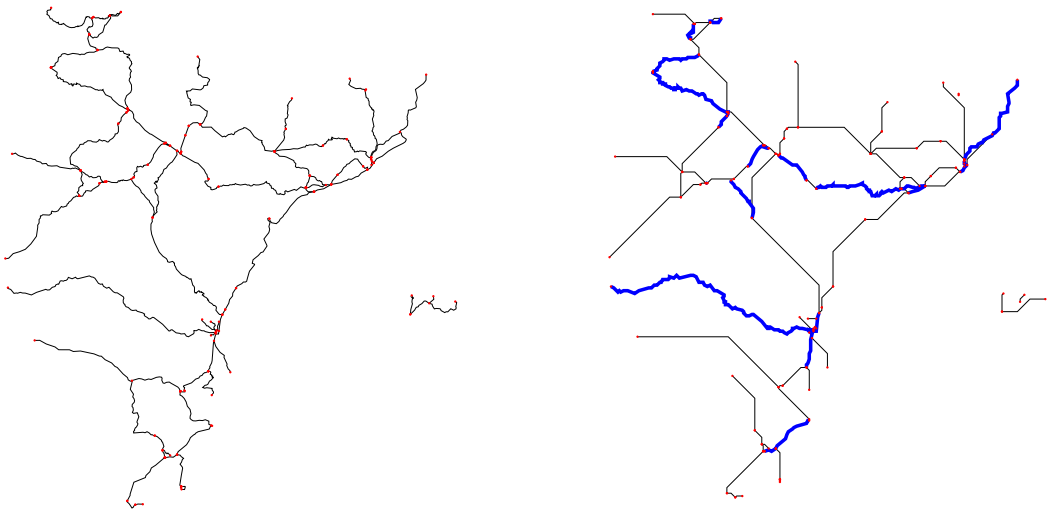

Figure 11: East of Spain. Left: original railway map. Right: schematic map with {HDH, VDV}-paths and without shared departure.

# 6 Conclusions and future work

This paper describes an efficient approach to compute schematized maps. Contrary to previous methods, our algorithm uses a bounded number of links per path, each of a given orientation (horizontal, vertical, or diagonal), as commonly used in transportation maps. Our approach, in contrast to previous works, is not an iterative, hopefully converging process, but it uses a combinatorial approach and always gives a correct solution if one exists. Experimental results that show the quality of the output have been presented.

There are many directions for further research. Firstly, we would like to be able to disallow shared departure for paths leaving the same endpoint in all cases, without sacrificing efficiency. This appears to be difficult for the vertical upward direction, if we want to guarantee a solution when one exists. Secondly, there are several types of schematic paths that cannot be handled by our method. It would be interesting to develop efficient algorithms that find a result if one exists in these cases too. Finally, before a schematization algorithm like ours can be useful in practice, certain local improvements and display styles must be incorporated, and more experimental work in order to improve the aesthetic quality of the output has to be done. Depending on the requirements for an aesthetic schematic map that follows all cartographic rules, this may involve more complex versions of the problem than studied in this paper.

# References


[1] http://www.cs.uu.nl/archive/sw/schematic-map/.

[2] M. Agrawala and C. Stolte. Rendering effective route maps: Improving usability through generalization. In *Proc. SIGGRAPH 2001*, pages 241–250, 2001.

[3] S. Avelar and R. Huber. Modeling a public transport network for generation of schematic maps and location queries. In *Proc. 20th Int. Cartographic Conference*, pages 1472–1480, 2001.

[4] S. Avelar and M. Müller. Generating topologically correct schematic maps. In *Proc. 9th Int. Symp. on Spatial Data Handling*, pages 4a.28–4a.35, 2000.

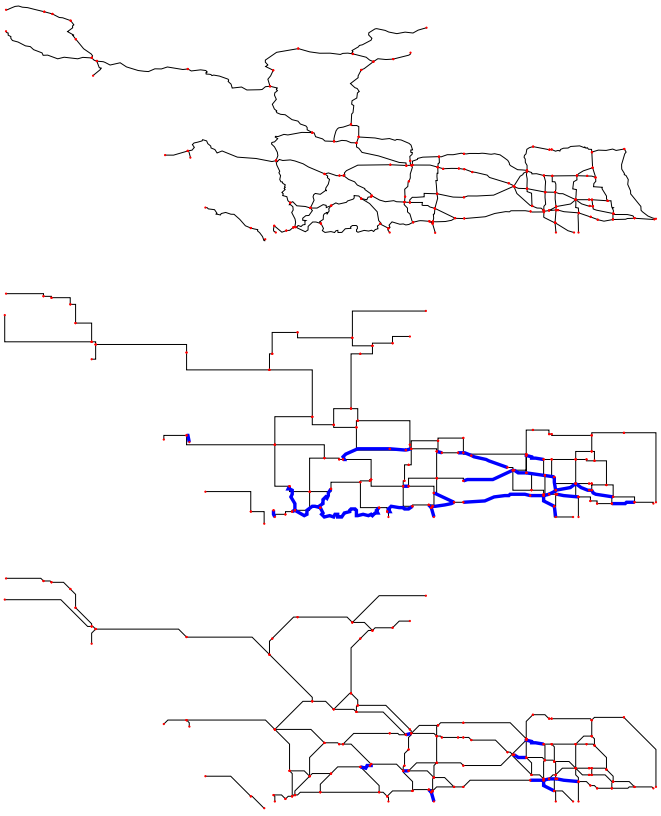

Figure 12: West of Canada. Top: original road map. Center: schematic map with two links per path without shared departure. Bottom: schematic map with {HDH, VDV}-paths without shared departure.


[5] T. Barbowsky, L.J. Latecki, and K. Richter. Schematizing maps: Simplification of geographic shape by discrete curve evolution. In *Spatial Cognition II, LNAI 1849*, pages 41–48, 2000.

[6] S. Cabello, Y. Leo, A. Mantler, and J. Snoeyink. Testing homotopy for paths in the plane. In *Proc. 18th Annu. ACM Sympos. Comput. Geom.*, pages 160–169, 2002.

[7] B. Chazelle. An algorithm for segment-dragging and its implementation. *Algorithmica*, 3:205–221, 1988.

[8] F.H. Croom. *Basic Concepts of Algebraic Topology*. Springer Verlag, Berlin, 1978.

[9] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing*. Prentice Hall, Upper Saddle River, NJ, 1999.

[10] A. Efrat, S.G. Kobourov, and A. Lubiw. Computing homotopic shortest paths efficiently. In *10th Annual European Symposium, ESA'02*, volume 2461 of *LNCS*, pages 411–423, 2002.

[11] D. Elroi. Designing a network line-map schematization software-enhancement package. In *Proc. 8th Ann. ESRI User Conference*. http://www.elroi.com/fr2_publications.html, 1988.

[12] D. Elroi. Gis and schematic maps: A new symbiotic relationship. In *Proc. GIS/LIS'88*. http://www.elroi.com/fr2_publications.html, 1988.

[13] S.H. Gerez. *Algorithms for VLSI Design Automation*. John Wiley & Sons, Chichester, 1999.

[14] A. Hatcher. *Algebraic Topology*. Cambridge University Press, 2001.

[15] F.M. Maley. Testing homotopic routability under polygonal wiring rules. *Algorithmica*, 15:1–16, 1996.

[16] G. Neyer. Line simplication with restricted orientations. In *Algorithms and Data Structures, WADS'99*, volume 1663 of *LNCS*, pages 13–24, 1999.

[17] L. Palazzi and J. Snoeyink. Counting and reporting red/blue segment intersections. *CVGIP: Graph. Models Image Process.*, 56(4):304–311, 1994.

[18] R. Raghavan, J. Cohoon, and S. Sahni. Single bend wiring. *J. Algorithms*, 7:232–257, 1986.

[19] R. Tamassia. On embedding a graph in the grid with the minimum number of bends. *SIAM J. Comput.*, 16(3):421–444, 1987.