Schematic Networks: An Algorithm and its Implementation [1]

Sergio Cabello and Marc van Kreveld

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

Abstract

This paper addresses research issues and describes a prototype implementation for the computation of schematic maps. The generated layout mainly depends on two parameters, namely the shape of the schematic connections and the distance between connections. The quality of the results as well as different issues regarding its construction are discussed. Additionally several extensions that can be implemented leading to an improvement of the final layout of the schematic maps and related applications are suggested.
Keywords: network, schema, map, visualisation

1 Introduction

Most underground systems in cities and most railroad companies show their maps with highly schematised connections between the major stations. The positions of these stations are usually preserved (approximately), but the shape of each connecting route or connection has been reduced to a polygonal line with only a few links (straight line segments) that are either horizontal, vertical, or diagonal (45° or 135°). The main advantage of such maps is that they provide a quick overview of the layout of the network, without showing unnecessary information like the precise shapes of the connections.

Several earlier papers discussed the automated construction of schematic maps (Elroi 1988a, Elroi 1988b, Elroi 1991), but the topic has reappeared lately in more papers (Avelar & Müller 2000, Barbowsky et al. 2000). All of these papers make use of some local operator, such that after applying it several times, the map converges to a schematic one. Different criteria based on different measures are used to decide in which order, and to which parts of the map the operator should

[1] This research is supported by the dr.ir. Cornelis Lely Stichting.

be applied. Iterative approaches like these can have prohibitive costs in time in interactive situations, and in those papers, no theoretical time bounds, nor actual computation times are given. Even convergence has not been shown formally.

A related topic that is relevant to schematisation is the rendering of particular routes under queries (Avelar & Huber 2001, Agrawala & Stolte 2001). Although the topic has its own characteristics, there are several common aspects.

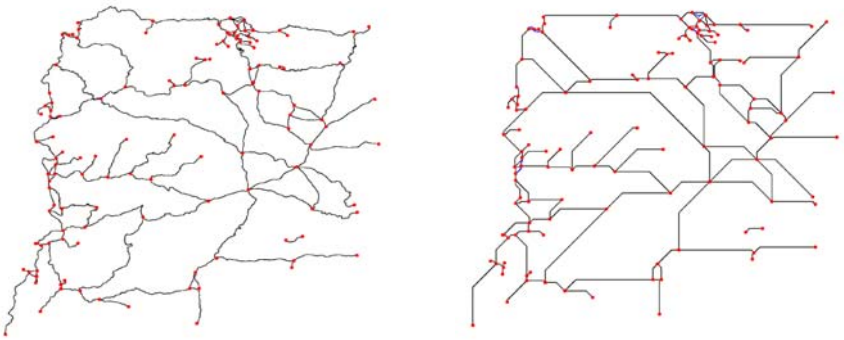
In a recent paper (Cabello et al. 2001), a relatively simple and efficient algorithm to produce schematised maps was presented. Its efficiency comes from the fact that it is combinatorial, in the sense that it does not apply methods to make the map converge slowly to a schematised one. Instead, it does some pre-computation and then places the connections immediately at their final positions. The algorithm is set up to obey a given set of requirements that are needed for the algorithm to work. There are also some choices the user can make for different visualisation styles. Then the algorithm produces a schematised map, if one exists, with the given requirements and choices. These are:

- Junctions of connections are not displaced.
- The allowed types of schematic connections can be chosen beforehand, but are fixed in the algorithm (schematisation models).
- Shared departure of different connections from the same junctions may be allowed or disallowed.
- A minimum separation distance between two connections may be chosen (not taking into account the first link of connections from the same junction).

The theoretical and algorithmic aspects of the algorithm were studied and proved in the aforementioned paper. This paper describes the implementation, results of experiments, and several practical aspects that were not previously addressed. The research questions of this paper are:

- Given the requirements above, and a schematisation model, how many connections (what percentage) can be placed for different data sets?
- How fast is the algorithm in practice?
- If not all connections can be placed, how should the algorithm deal with non-placeable connections?
- Can the algorithm be adapted to incorporate through-going connections at junctions, that is, to enforce that some connection is the prolongation of some other connection at a junction?
- How should the algorithm deal with very short connections?
- How should the minimum separation distance be implemented?
- Can obstacles (other map features) be incorporated in the algorithm?
- Can the output of the algorithm be improved aesthetically, without sacrificing the number of placed connections?
- Can the minimum separation distance be maximised, without placing fewer connections?

These questions are addressed by developing methods that extend the algorithm of Cabello et al. (2001), and provide a prototype implementation in Java which can be tested at the web page http://www.cs.uu.nl/archive/sw/schematic-map/. Three different data sets with railways and roads of countries were chosen as test data. The most important conclusion is that the algorithm is fast in practice, places nearly all connections, and gives topologically correct, aesthetic maps as in the output for Fig. 1.



(a) (b)

Fig. 1. Northwest of Iberic Peninsula, **(a)** original map after editing with the prototype, **(b)** the schematised version made by the prototype

This paper begins with a global description of the algorithm of Cabello et al. (2001), which is needed to understand its features and limitations. The implementation prototype and its possibilities are then described. Section 4 gives the results obtained from experiments on different data sets, with different settings of the implementation. Further visualisation possibilities of the method are presented followed by the conclusions.

2 The Algorithm and its Parameters

In this section, an explanation is provided concerning the fundamental ideas about the algorithm of Cabello et al. (2001). Assume a network or map is presented, such as a railway map, and the desired type of schematic connections to be used, such as the 3-link connection type are specified. In brief , the algorithm is as follows: first a top-to-bottom ordering of the connections of the original map is computed, and then the schematic connections are placed, one by one, following this order. As much freedom as possible is left for connections to be placed later. The concepts and ideas are explained in the following sections..

Imagine a physical model of the original map where each connection is made of an elastic cord with fixed endpoints. One natural approach to construct the

schematic map could be to continuously deform some connections in order to make them more schematic and simple whilst keeping the topology of the whole network. This process can be continued until some schematisation level is reached. A similar idea is used in Avelar & Müller (2000), where the positions of the junctions and important points are changed iteratively to create a schematisation. With this type of approach, the schematic map keeps the topological relations from the original map, as well as some spatial relations. For example, the cyclic order of the roads leaving one city is as in the original map.

Some observations on this approach imply the need for a combinatorial algorithm. As a rule, when a connection is deformed, as described, it is useful to provide as much freedom as possible for the next deformation. If, for example, one connection in the centre of the map is deformed, its schematised placement will limit the possibilities of placement of other connections both above and below it. Hence, it is not possible to decide which is the best possible deformation. If, however, the connection at the top of the map is deformed and it is deformed such that it leaves as much space as possible below it, will then influence later deformations in a limited manner. This process is therefore continued in which the top-most connections are deformed among the ones not yet schematised.

When deforming one connection to leave as much space as possible for later, all connections below it play no role (that is, do not influence its placement). All connections already placed above it, however, have to be observed, otherwise the topology of the network could be changed. This brings us to the final structure of the algorithm:

- Extract all connections from the original map and order them from top-to-bottom in a list.
- Remove all connections from the original map, but keep the endpoints where they are.
- Place, one by one and leaving as much space as possible for later, each connection from the ordered list in the map. Each newly placed connection must be of the desired shape and below the previously placed connections.

To order the connections a decision must be reached for each pair of connections, with respect to their above-below relation. Observe that this order relation is computed in the original map and will be preserved in the schematic map, so it is an invariant relation under continuous deformations of the connections. For this reason, one point P is considered to be below a connection c if the upper vertical half-line from P intersects any continuous deformation of the connection c that does not pass over P. To decide if P is above is done in the same way but using the lower half-line. It can also be the case that any deformation of c intersects the half-line above P and the half-line below P, in which case P is both above and below c. Some cases are shown in Fig. 2.

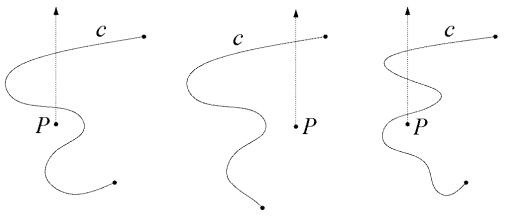


Fig. 2. Left: *P* is neither above nor below *c*. Center: *P* is below *c*. Right: *P* is below and above *c*

To determine the above-below relation of two connections the endpoints of one connection are compared with the other connection, and vice versa. Details of the efficient computation of the ordering are given in Cabello et al. (2001).

In the algorithm, for placing the next connection, some considerations have to be taken into account. For example, to facilitate the legibility, the schematic connections must retain some distance between them. Observe that the distance between two connections depends on whether they are sharing an endpoint or not (see Fig. 3). When two connections do not share any endpoint, the distance is the minimum over all the vertical distances from any link (edge) to the other connection. If two connections share an endpoint, the minimum must be taken, but without taking into account the distances from the first link to the other connection. The reason to use only the vertical distance is due to the fact that an ordering of the connections from top to bottom is used

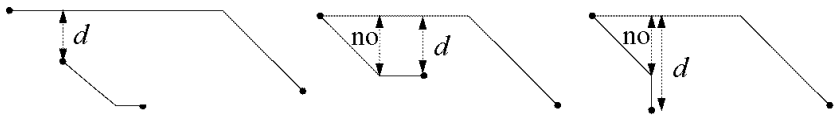
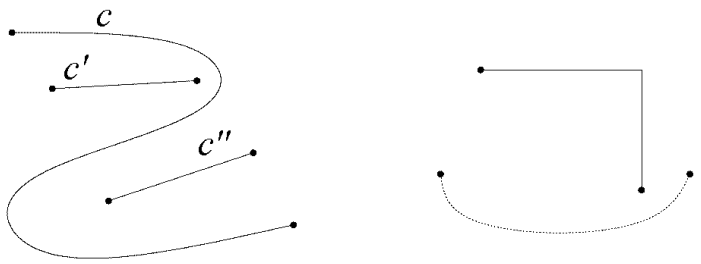


Fig. 3. The distance between connections depends on whether they share endpoints or not. The distances labelled "no" are not considered

The algorithm can fail in two different ways. When computing the order, it can happen that the connections cannot be ordered. For example, the left part of Fig. 4 gives an example where no connection is topmost, since each one has another connection above it. The algorithm can also fail because the next place of the connection cannot be placed. For example, to the right in Fig. 4, use is restricted to the 2-link orthogonal (axis-aligned) schematic connections. As a result the dashed connection cannot be placed without intersecting the other one.



(a) (b)

Fig. 4(a) the connections cannot be ordered: *c* above *c'*, *c'* above *c''*, and *c''* above *c*, **(b)** the schematised version of the dashed connection cannot be placed if the orthogonal 2-link connections are restricted

Although the idea of the algorithm can be applied to any type of schematic connection, certain statements can be proved in the following cases:

- 2-link orthogonal;
- 3-link: orthogonal, diagonal, orthogonal;
- 3-link: diagonal, orthogonal, diagonal;

where all of them have to be shortest in their class.

Theorem: For the above types of schematic connections, there is an algorithm that, in $O(n \log n)$ time, schematises any map with n segments. Furthermore, if the algorithm fails to report a schematic map, then no schematic map with the desired connection type exists.

3 The Prototype

An experimental prototype using the ideas above has been implemented as a Java applet. Although some techniques to speed up the algorithm that were explained in Cabello et al. (2001), are not used, the processing time does not appear to be problematic.

Editing tools were included in the prototype in order to make it more flexible and interactive for the user. Observe that the algorithm described above does not move the endpoints of connections, but some relocation of them may be desired, so this tool can be important. For example, it could be useful to identify and merge the endpoints of connections when they are very close in the schematic map. Observe that when two connections have a slight overlap in the vertical direction and they are closer in the vertical direction than allowed, that this algorithm would not schematise the lower one, see Fig. 5. In addition, the presence of intersections close to endpoints plays an important role, since problems arise with ordering the connections. Sometimes such problems can be resolved by editing, see Fig. 5.

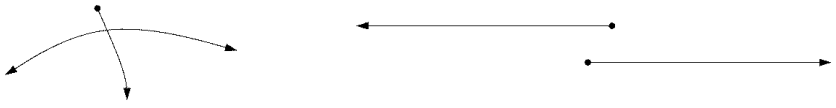


Fig. 5. Some data may produce problems if it is not edited, **(a)** intersections may produce a problem to decide which connection is above. Thus, it must be topolgically correct. **(b)** If two endpoints are closer than the minimum allowed distance, and they are not identified, the lower connection cannot be schematised.

The prototype has four different screens, each one referring to a different map (see Fig. 6).

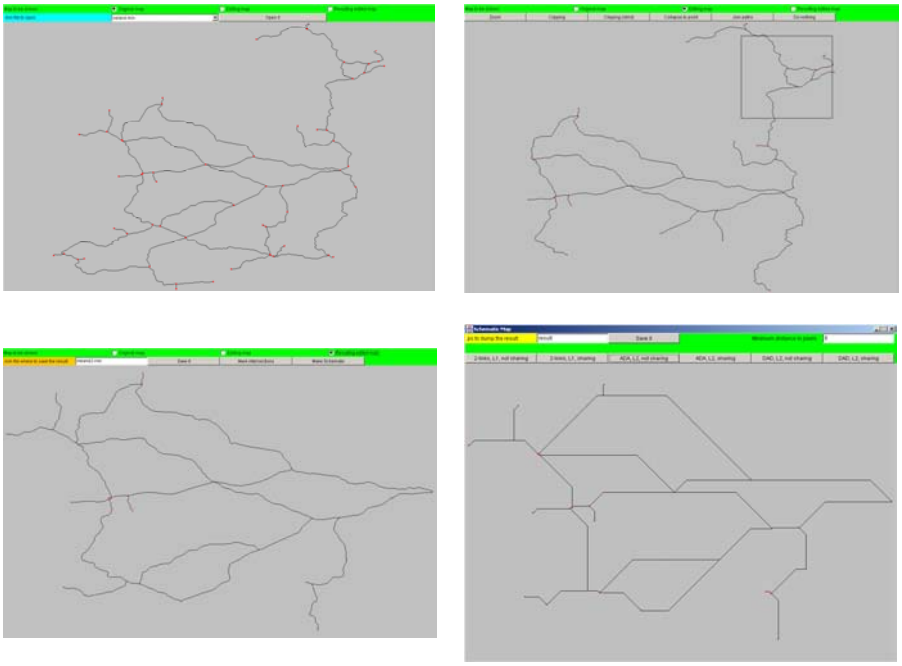


Fig. 6. Four screenshots of the prototype

- The first screen (original map) is used for opening the file with the data. This map cannot be changed and it remains the same all the time.
- The second screen (editing map) allows to select a rectangular region from the map and perform some operation on that region:
  - Zoom on that region;
  - Clip: selects all connections partially in the rectangle;
  - Clip (strict): selects all connections lying completely in the rectangle;

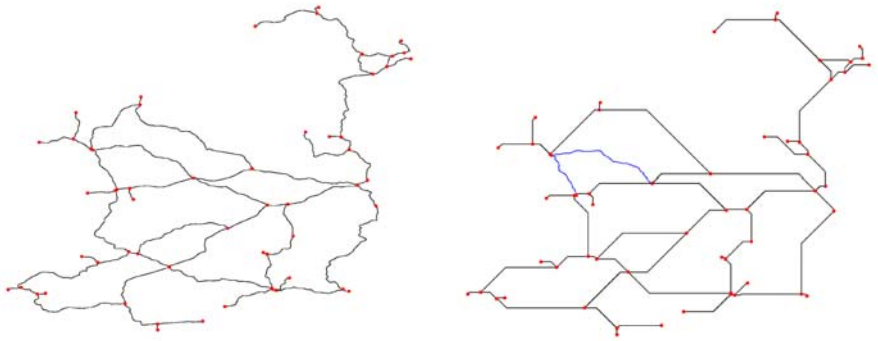
  - Collapse to point: moves all endpoints and junctions in the rectangle to the centre of it;
  - Join connections: if there are only two endpoints in the rectangle, both incident to only one connection, it joins the corresponding connections to one new connection (the two endpoints merge and become an intermediate vertex of the connection). If there are several connections inside the rectangle, it merges any two connections incident to some endpoint by turning it into an intermediate vertex;
  - Do nothing: do not make any change.
- The third screen is used to visualise the editing results, which includes buttons to compute the intersections of the connections. As this is an expensive operation for maps with many connections, it is recommended - after using it - to save the map such that it does not have to be repeated. Pressing the button "Make schematic" opens a new window in which the schematic map will be made.
- In the schematic map window (fourth screen), the desired minimum distance is set between the schematic connections. As a result, the different types of schematic connections can then be selected, which are as follows;
  - 2-link orthogonal;
  - 3-link, with the first and last link orthogonal, and the middle one diagonal. This connection has to be shortest given this type.
  - 3-link, with the first and last link diagonal, and the middle one orthogonal. This connection has to be shortest given this type.

The input data is organised as a collection of connections, each connection being a collection of points. A format (MIN) was created for reading this data, as well as a converter from ESRI shape (.shp) files to the format (MIN). The results can be saved as MIN or Postscript files depending on the extension used (.min or .ps).

The order among the connections is computed when opening the window that shows the fourth screen. So it is computed only once. In this window, the minimum distance can be specified. Because it is a visualisation parameter, its units are pixels. The schematic map is then computed, where the connections that could not be schematised are shown in blue in their original shape. A small modification to improve the result was introduced in the algorithm> it computes a schematic map M using the specified minimum distance d; *d is then increased* to find the biggest value that produces a schematic map with the same number of schematised connections as M. The schematised map with those settings is the output map. The search for the best value of d is done by a binary search, so that not all values for d have to be tried incrementally.

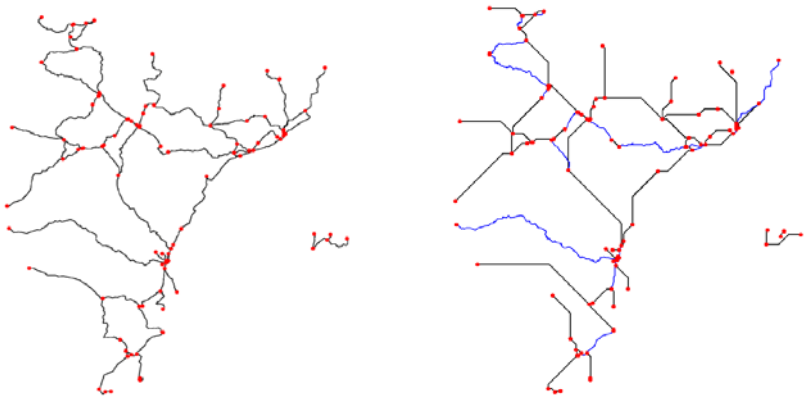
4 Experimental Results

The prototype was used with maps from Canada, Ireland and Spain, the number of connections per map are given in Table 1. In all of them, the time of computation is within 3 seconds. However, the most time consuming part is the binary search to find the optimal distance, as explained at the end of the previous section. In all examples and results in this section, the smallest allowed vertical distance between two connections was set to 10 pixels. Some of the results can be seen in Figs. 7-9.



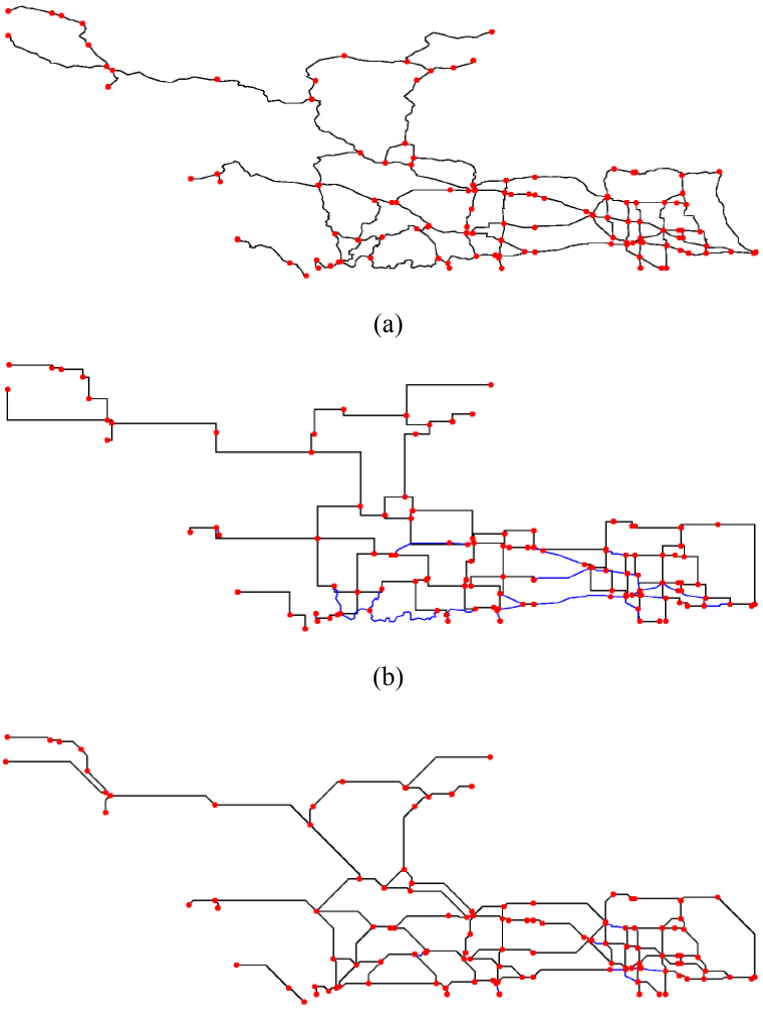
(a) (b)

Fig. 7. Ireland, **(a)** original railway map, **(b)** schematic map with (orthogonal, diagonal, orthogonal) connections and without sharing departure



(a) (b)

Fig. 8. East of Spain, **(a)** original railway map, **(b)** schematic map with (orthogonal, diagonal, orthogonal) connections and without sharing departure



(a)

(b)

(c)

Fig. 9. West of Canada, **(a)** original road map, **(b)** schematic map with two links per connection without sharing departure, **(c)** schematic map with (orthogonal, diagonal, orthogonal) connections without sharing departure

For the maps of Ireland and Canada, the top-to-bottom placement order was computed without any problem. Problems were encountered when computing among the connections for the map of Spain. This was due to inaccuracies in the data, which is topologically not correct. In particular, in the part shown in Fig. 8, there were intersections between different connections. After computing and resolving those intersections, the placement order was computable.

As can be seen in the pictures, some connections cannot be placed because they would intersect some connection already placed, they would give sharing departure, or they would not keep the proper distance. Table 1 shows the connections that were present in the original data, and the percentage of them can be placed without and with sharing departure. As can be seen, the improvement is only small when sharing departure is allowed. After analysing the pictures in the prototype, which allows zooming in the original map, one can see that the biggest source of problems is that the endpoints close together were not identified and merged. So, most of the non-schematised connections would have been schematised after some editing of the map beforehand.

Table 1. Percentage of placed connections for the maps shown in this article

	Number of connections	Not sharing departure	Sharing departure
Fig. 7 right	65	93.8 %	93.8 %
Fig. 8 right	100	76 %	79 %
Fig. 9 centre	155	74.1 %	86.4 %
Fig. 9 bottom	155	89.6 %	89.6 %

5 Other Visualisation Issues

Some features can be listed that improve the visualised results. Most of them could be included in the prototype in a straightforward manner, however, this has not been done yet. These extensions are:

- Obstacles can be managed when the connections keep the same spatial relation with them (passing above or below). For example, when the obstacles are points, they can be treated as connections of length zero in the algorithm.
- The same algorithm can be applied to boundaries of countries or administrative maps as well (for instance, for choropleth maps). For this purpose, the boundaries should be split into a few pieces.
- Usually, sharp corners do not look nice, and the visualisation can be improved by rounding them.
- Sharing departure from one endpoint can be avoided by representing the endpoints by rectangles. The connections can then leave the rectangle as a bundle of parallel links. With a few adaptations, it is possible to produce schematic maps that are commonly used for subway lines in cities.
- When some endpoint represents the junction of two or more major transportation lines, it is desirable to have continuity of the connections of the same line, where one is the prolongation of the other one on opposite sides of the junction. In some of the cases, this can be handled with a bottom-up post processing in the schematic map.
- After computing the order among the connections, other ways of placing the connections, such as alternating between the topmost and the bottommost

connection, may give better results. Here, the bottommost connections should be placed in the bottommost position. This improvement avoids the risk of a cushioning effect at the bottom of the map. In the examples, however, such effects are not apparent even without this adaptation.

6 Conclusions and Further Research

In this paper the algorithmic approach of Cabello et al. (2001), has been presented and discussed through the review of a prototype implementation. Several extensions to that paper, which improve the visualisation, have been incorporated, and also editing tools to obtain topological consistency of the input map. Experiments show that most problems can be resolved with some editing of the data, especially using identification of junctions and the merge operator The prototype presented in this paper can be accessed and used via Internet at http://www.cs.uu.nl/archive/sw/schematic-map/.

The algorithm presented (Cabello et al. 2001) is the first method to take a combinatorial approach to schematic map construction, and as such is very efficient. This paper shows that the method is fast, and the approach is versatile. In addition, various adaptations and extensions allow the visualisation of other styles of schematic maps, including subway maps and schematised choropleth maps.

References

Agrawala M, Stolte C (2001) Rendering effective route maps: Improving usability through generalisation. In: Proc. Siggraph 2001. pp 241-250

Avelar S, Huber R (2001) Modeling a public transport network for generation of schematic maps and location queries. In: Proc. 20th Int. Cartographic Conference. pp 1472–1480

Avelar S, Müller M (2000) Generating topologically correct schematic maps. In: Proc. 9th Int. Symp. on Spatial Data Handling. pp 4a.28–4a.35

Barbowsky T, Latecki L J, Richter K-F (2000) Schematizing maps: Simplification of geographic shape by discrete curve evolution. In: Spatial Cognition II, LNAI 1849. pp 41–48

Cabello S, de Berg M, van Dijk S, van Kreveld M, Strijk T (2001) Schematization of road networks. In: Proc. 17th Annu. ACM Sympos. Comput. Geom. pp 33–39

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

Elroi D (1988b) GIS and schematic maps: A new symbiotic relationship [online]. In: Proc. GIS/LIS'88. Available from: http://www.elroi.com/fr2_publications.html

Elroi D (1991) Schematic views of networks: Why not have it all. In: Proc. of the 1991 GIS for Transportation Symposium. pp 59–76, Available from: http://www.elroi.com/fr2publications.html