Research Article

Automated Production of Schematic Maps for Mobile Applications

J. Mark Ware  
School of Computing  
University of Glamorgan

Suchith Anand  
School of Computing  
University of Glamorgan

George E. Taylor  
School of Computing  
University of Glamorgan

Nathan Thomas  
School of Computing  
University of Glamorgan

Abstract
The advent of high-end miniature technology, together with the increasing availability of large scale digital geographic data products, has created a demand for techniques and methodologies that assist in the automated generation of maps specifically tailored to mobile GIS applications. This paper concerns itself with the problem of automatic generation of schematic maps. Schematic maps are diagrammatic representations based on linear abstractions of networks. In the context of mobile mapping they are seen as being a particularly useful means of displaying transportation networks. This paper describes an algorithm that automates the production of schematic maps. The algorithm makes use of the simulated annealing optimisation technique. An implementation of the algorithm is also presented, together with experimental results.

1 Introduction

A schematic map is a diagrammatic representation based on linear abstractions of networks. Typically transportation networks are the key candidates for applying schematization. Perhaps the most well known example of a schematic map is the London Tube map designed by Harry Beck in 1931 (see http://www.tfl.gov.uk/tfl/maps-home.shtml for this and many hundreds of other examples). An electrical engineer, Beck based his design on a circuit diagram and used a schematic layout. His map locally distorted the scale and shape of the underground rail route but preserved the overall topology of the rail network.

Address for correspondence: J. Mark Ware, School of Computing, University of Glamorgan, Pontypridd CF37 1DL, Wales, UK. E-mail: jmware@glam.ac.uk
The map is further enhanced by colour coding lines and using intuitive icons for interchange stations.

Schematic maps are built-up from sketches that usually have close resemblance to verbal descriptions about spatial features (Avelar 2002). Due to their abstracting power, schematic maps are an ideal means for representing specific information about a physical environment. They play a helpful role in spatial problem solving tasks such as way finding. While topographic maps are intended to represent the real world as faithfully as possible, schematic maps are seen as conceptual representations of the environment. Schematic maps provide a suitable medium for representing meaningful entities and spatial relationships between entities of the represented world (Casakin et al. 2000). Differing geometric and aesthetic criteria can be used to design a schematic map keeping in mind the common goals of graphic simplicity, retention of network information content and presentation legibility (Avelar and Müller 2000).

Generating a schematic map involves reducing the scale and complexity of map details while at the same time preserving the important characteristics. It can be regarded as a type of map generalization, which is loosely defined as the process by which small scale maps are derived from large scale maps. If performed manually, map generalization is a time-consuming and expensive process. Automating the process has been, and continues to be, the subject of much research (e.g. Weibel 1995, Weibel and Jones 1998, Jones and Ware 2005). The advent of high-end miniature technology, together with the now widespread availability of relatively large-scale digital geographic data products, has created the particular need for automated map generalization solutions tailored specifically to mobile GIS applications. This paper presents one such solution in the form of an algorithm for automatically generating schematic maps from large scale network data. In the context of mobile mapping, the authors see schematic maps as being a particularly useful means of displaying routes for wayfinding type applications.

Mobile GIS is a relatively new technology, but with the availability of digital geographic datasets its application potential has increased tremendously. There is a huge amount of available geographic information that can be re-purposed for mobile GIS applications; together with the ability to filter and personalize content by reference to a user’s physical location, this will provide compelling business and research opportunities in this emerging field.

The main components of a mobile GIS are a positioning system (such as GPS), a handheld computer and a communication network, with GIS acting as the backbone (Figure 1). To understand the unique issues related to generalizing maps for display on handheld computers devices it is important to consider the device specifications and tailor the solutions to suit the need. For example, the current specification (in terms of processing power, memory capacity, screen resolution and display size) of a typical Personal Digital Assistant (PDA) is very much less than that of a typical personal computer. Hence there is a marked difference between the map generalization requirements for display on a contemporary desktop computer and those of a thin client mobile device, like a PDA. For example, the processing capability of contemporary PDAs is in the range of 400 MHz. Their memory capacity is in range of 64 MB. This highlights issues associated with processing and storage of large-scale voluminous datasets in thin client mobile devices. In addition, the low display resolution of 240 × 320 pixels and smaller display area (50 cm² on a PDA) makes it necessary to render the final output image based on generalization carried out as per appropriate small display cartographic specification to give maximum clarity and readability. The basic criteria on which such specification...
must be based are easy readable font, recognizable symbols, mutual exclusive colour at each level of information and comprehensive use of area colour with few geometric details of objects (GiMoDig Project 2003). In summary, PDAs have different form factors such as display resolution, varying numbers of display lines, horizontal or vertical screen orientation and hardware specification when compared to contemporary desktop computers. Hence GIS applications that are to be used in PDAs need to be tailored appropriately. The application of suitable automated map generalization techniques will help in filtering redundant data. This will enable faster and more efficient rendering, and reduce noise and enhance essential detail in the rendered image (Anand et al. 2004).

The remainder of the paper is set out as follows. Section 2 outlines the key generalization processes involved in the production of schematic maps, with reference to previous automated solutions to the problem. Section 3 contains a description of the simulated annealing-based schematic map generator algorithm that forms the basis for this paper. A prototype implementation of this algorithm is described in section 4, and some experimental results are presented. The paper concludes in section 5 with a summary of the results and a discussion of future work.

2 Schematic Map Production

Waldorf (1979) defines the basic process for generating schematic maps as involving the elimination of all networks (or portions of networks) and features that are not functionally relevant to the single system chosen for mapping. All geometric invariants of the network’s structure are relaxed except topological accuracy. Routes and junctions are symbolized abstractly.

Elroi (1988) refined the process by adding three graphic manipulations, although implementation details and results were not given. First, lines are simplified to their most elementary shapes. Next, lines are re-oriented to conform to a regular grid, such that they all run horizontally, vertically or at a forty-five degree diagonal. Third, congested areas are increased in scale at the expense of scale in areas of lesser node density. These three steps are illustrated in Figure 2.

The first step in the process is line simplification, which can be achieved using an algorithm such as that of Douglas and Peuker (1973). Care must be taken when performing this step to avoid the introduction of topological errors; this can be achieved most easily by making use of topology preserving variants of the Douglas-Peuker algorithm, such as that presented by Saalfeld (1999).

Steps two and three are the key components of the process, and their automation has been the focus of previous work by several researchers. Avelar and Müller (2000)
present an algorithm for the automatic generation of schematic maps from traditional vector-based route networks, which are pre-generalized using the Douglas-Peucker algorithm. They make use of gradient-descent based optimization in an attempt to force the network to conform to orientation and minimum separating distance constraints. Map modifications are achieved by the iterative displacement of map vertices. At each iteration, vertex displacements are calculated and, provided topological consistency is maintained, applied. Cabello et al. (2001) present a combinatorial algorithm for the schematization of road networks. The algorithm produces maps in which every path has two or three links and edge orientation is restricted (horizontal, vertical or diagonal). It is guaranteed to find a correct solution if one exists, but has the disadvantage of not providing an output if such a solution does not exist.

3 A Simulated Annealing Approach to Generating Schematic Maps

Simulated annealing (SA) (Kirkpatrick et al. 1983) is a probabilistic heuristic optimisation technique used for finding good approximate solutions to the global optimum of a given function in a large search space. SA is analogous to the way in which a metal cools from an initially high temperature until it freezes into a minimum energy crystalline structure (called the annealing process). It has been used as an optimization tool in a wide range of application areas, including routing, scheduling and layout design (e.g. Cerny 1985, Elmohamed et al. 1998, Chwif et al. 1998), including automated cartographic design (Zoraster 1997, Ware et al. 2003). In this paper, the schematization process is considered as an optimization problem. Given an input state (a network layout), an alternative state
can be obtained simply by displacing one or more of the network vertices. The search space being examined is therefore the set of all possible states of a given input linear network. Each state can be evaluated in terms of how closely it resembles a schematic map. However, finding the best state by exhaustively generating and evaluating all possible states is not possible, as for any realistic data set the search space will be excessively large (i.e. there are too many alternative layouts). SA offers a means by which a large search space can be searched for near optimal solutions. A standard SA algorithm, which is adopted for use in this work, is shown in Figure 3 and explained below.

3.1 Simulated Annealing Algorithm

An SA-based schematization algorithm is now presented. The approach used is similar to that used by Agrawala and Stolte (2001) to produce sketch route maps. At the start of the optimisation process SA is presented with an initial approximate solution (or state). In the case of the schematic map problem, this will be the initial network (line features, each made up of constituent vertices). The initial state is then evaluated using a cost function $C$; this function assigns to the input state a score that reflects how well it measures up against a set of given constraints. If the initial cost is greater than some user defined threshold (i.e. the constraints are not met adequately) then the algorithm steps into its optimisation phase. This part of the process is iterative. At each iteration the current state (i.e. the current network) is modified to make a new, alternative approximate solution. The current and new states are said to be neighbours. The neighbours of any given state are generated usually in an application-specific way. In the algorithm presented here, a new state is generated by the function RandomSuccessor, which works by selecting a vertex at random in the current state and subjecting it to a small random displacement. The new state is also evaluated using $C$. A decision is then taken as to whether to switch to the new state or to stick with the current. Essentially, an improved new state is always chosen, whereas a poorer new state is rejected with some probability $p$, with $p$ increasing over time. The iterative process continues until stopping criteria are met (e.g. a suitably good solution is found or a certain amount of time has passed).

Figure 3 Simulated annealing algorithm
3.2 Probability $p$

At each iteration the probability $p$ is dependant on two variables: $\Delta E$ (the difference in cost between the current and new states); and $t$ (the current temperature). $p$ is defined as:

$$p = e^{-\Delta E/t}$$  \hspace{1cm} (1)

where $t$ is assigned a relatively high initial value; its value is decreased in stages throughout the running of the algorithm. At high values of $t$ higher cost new states (large negative $\Delta E$) will have a relatively high chance of being retained, whereas at low values of $t$ higher cost new states will tend to be rejected. The acceptance of some higher cost new states is permitted so as to allow escape from locally optimal solutions. In practice, the probability $p$ is tested against a random number $r$ ($0 \leq r \leq 1$). If $r < p$ then the new state is accepted. For example, if $p = 1/3$, then it would be expected that, on average, every third higher cost new state is accepted. The initial value of $t$ and the rate by which it decreases is governed by what is called the annealing schedule. Generally, the higher the initial value of $t$ and the slower the rate of change, the better the result (in cost reduction terms); however, the processing overheads associated with the algorithm will increase as the rate of change in $t$ becomes more gradual.

3.3 Constraints

The viability of any SA algorithm depends heavily on it having an efficient cost function, the purpose of which is to determine for any given element of the search space a value that represents the relative quality of that element. The cost function used here, $C$, is called repeatedly and works by assessing the extent to which a given state meets a set of constraints. In the current version of the schematic map algorithm seven constraints are considered:

1. Topological – original network and derived schematic map should be topologically consistent;
2. Orientation – if possible, network edges should lie in horizontal, vertical or diagonal direction;
3. Length – if possible, all network edges should have length greater than some minimum length (in order to reduce congestion);
4. Angle – if possible, the angle between a pair of connected edges should be greater than some minimum angle;
5. Rotation – an edge’s orientation should remain as close to its starting orientation as possible;
6. Clearance – if possible, the distance between disjoint features should be greater than some minimum distance;
7. Displacement – vertices should remain as close to their starting positions as possible.

Each of these constraints, illustrated in Figure 4, can be evaluated using straightforward computational geometry functions (e.g. edge/edge intersect test and vertex to edge distance calculation). In order to work efficiently, certain of these functions require the use of a spatial index of some kind. In the current implementation, a simple regular grid indexing scheme is used. When invoked initially, $C$ evaluates a cost for each feature in the network. This cost represents the extent to which each feature meets the set of constraints. The overall cost is found by summing the individual feature costs. A record of the individual
feature costs is maintained for future reference, meaning that, in any further call, C has to consider only features with costs affected by the most recent vertex displacement.

It will usually be the case that a certain constraint is considered more important than others. For example, the topological constraint will almost always be deemed the most significant, while orientation will usually have a higher priority than the remaining constraints. Any such order of precedence amongst constraint types can be accommodated by weighting the cost associated with each type of constraint violation. In the implementation described in the next section a user can interactively set the weight of each constraint.

Figure 4  Constraints: (a) Topological – original (Left), topological error (Middle) and acceptable solution (Right); (b) Orientation – original (L) and schematized (R); (c) Length – original (L) and congestion reduced by enforcing Length constraint (R); (d) Angle – edges re-oriented but Angle constraint violated (L) and acceptable solution (R); (e) Rotation – original (L), acceptable solution (M) and better solution (R); (f) Clearance – constraint violated (L) and resolved (R); (g) Displacement – original (L), acceptable solution (M) and better solution (R)
3.4 Symbolization

In some of the example output shown in section 4, certain polylines (i.e. connected edges between nodes) in the map belong to more than one route. In order to deal with these situations, additional software has been developed to symbolize routes. The symbolization process makes use of a function OffsetPolyline, which produces a duplicate, but offset, version of an input polyline (see Figure 5 for example). The offset value, which can be either positive or negative, is an input parameter to the function. It is assumed that detailed explanation of this function is not necessary here and a VB version is included as an appendix to this paper. The function works well in most cases, but will struggle in situations where there is a sharp angle between a pair of connected edges. However, it this particular application such situations are unlikely to occur due to the inclusion of the Angle constraint.

4 Implementation and Experimental Results

Prototype software for producing schematic maps for transportation network data has been developed. It makes use of the SA algorithm described in the previous section, which is implemented as a VBA script within ArcGIS. The software has been tested on OSCAR road centre line data for the St. David’s area of West Wales. In the results shown in Figures 6 and 7, the software has been used to generate a series of schematic route maps corresponding to inputs supplied by a user. The schematic map is output in shapefile format. Any output polyline belonging to more than one route is processed using OffsetPolyline and the resulting offset polyline is added to the output shapefile. For example, if two routes lie on a particular polyline then two polylines are output – one is the result of applying a positive offset and the other is the result of applying a negative offset. If there are three routes then three polylines are output – the original schematized polyline plus its positive and negative offsets. Additional routes can be catered for in similar fashion.

4.1 Initial Results

The original test data (Dataset1) consisted of 124 line features, made up from a total of 748 vertices. This data is pre-generalized using the ArcInfo Generalize tool; with point
remove and topological error check options selected this makes use of an enhanced version of the Douglas-Peuker algorithm. A weed tolerance value of 30 (m) is used in these experiments, resulting in 124 line features, made up from a total of 264 vertices. The simplified data acts as input to the schematic map generator script. For the results

Figure 6  A sample schematic map produced using the simulated annealing software. TOP – Pre-generalized data (OSCAR, Ordnance Survey © Crown Copyright). BOTTOM – Schematic map generated by simulated annealing software and automatically symbolized
shown, the Topological constraint is given a very much higher weighting than all other constraints. This serves to ensure that topological consistency is maintained between input and output map (at the expense, in some instances, of the other constraints). The Orientation constraint is also given a relatively high weighting, while the other constraints (3) to (7) are given low weights. As can be seen, the algorithm has been successful, both in terms of alignment and in reducing congestion. The sample schematic map shown took less than 20 s to generate (using a 3 GHz Pentium 4 PC with 1 GB of RAM).

### 4.2 Douglas-Peuker Tolerance

Application of the Douglas-Peuker line simplification to a set of line features results in a new set of line features in which each feature is represented by a subset of its original vertices. The number of vertices removed during the process (i.e. the level of simplification) depends both on the complexity of the input data, and a user-defined parameter referred to as the weed tolerance. In general, the higher the value of this tolerance, the greater the number of vertices removed. In the datasets used in the experiments presented in this paper, a tolerance value of 30 m is used. This value was arrived at after visually inspecting schematics produced using a range of values. Figure 8 provides sample output. These experiments make use of Dataset2 (a sub-portion of Dataset1), which consists of 67 line features and a total of 371 vertices. It was found that tolerance values of above about 10 m produce good results. Values less than 10 m give schematics in which there is too much detail.
4.3 Annealing Schedule

A key factor determining the success or failure of simulated annealing is the choice of annealing schedule. If the simulated annealing process proceeds too quickly then poor quality results are likely; a process that runs very slowly might not be suitable for some applications. The annealing schedule approach adopted in this work is similar to the format of Christensen et al. (1995). This involves setting $t$ to an initial value $\tau$. At each temperature a maximum of $\omega n$ vertex displacements (successful or unsuccessful) are allowed, where $n$ is the total number of vertices in the map. After every $\omega n$ displacements $t$ is decreased geometrically such that $t_{\text{new}} = \lambda t_{\text{old}}$. In addition, if more than $\xi n$ successful displacements (i.e. the new state is accepted) are made at any one temperature then $t$ is immediately decreased. If no successful displacements are made at a particular temperature then the algorithm terminates. Finally, a limit on the maximum number of temperature stages allowed is set to $\psi$ (in practice the maximum number of temperature stages is never reached). The schedule parameter values $\tau$, $\lambda$, $\omega$, $\xi$, and $\psi$ are provided by the user at runtime. In the current implementation a good set of values is arrived at by trial and error.

Figure 8  Schematic maps produced from Dataset2 pre-generalized using a range of Douglas-Peuker tolerance values: (a) 0 m (371 vertices); (b) 1 m (254 vertices); (c) 2 m (205 vertices); (d) 10 m (150 vertices); (e) 30 m (142 vertices); and (f) 50 m (134 vertices)
For the datasets used in this paper the values \( \tau = 2, \, \lambda = 0.9, \, \omega = 30, \, \xi = 10, \) and \( \psi = 50 \) were chosen as they were found to provide a trade-off between efficiency and quality of solution. However, obtaining suitable parameters in this way is time-consuming; finding a way by which they can be set automatically will be the focus of future research.

4.4 Process Termination

As with similar optimisation approaches (such as Genetic Algorithms), an important aspect of the simulated annealing process is to know when to terminate. In the implementation described here, the process stops if any of four stopping criteria are met. These criteria are:

1. A zero cost solution has been found;
2. There has been no improvement for a user-defined number of iterations;
3. The number of temperature changes \( > \psi \); and
4. A user-defined period of time has elapsed.

In order to assist users in setting the input parameters for criterion (4), experiments, using Dataset1, were carried out to assess the relationship between schematic map quality (i.e. cost) and execution time. The results, which are summarised in Figure 9, show that approximately 90% of the total cost reduction is achieved during the first 4 or 5 seconds of optimisation; further significant reduction takes place during the next 20 or so seconds, with very limited improvement made thereafter. This characteristic of the process is further illustrated in Figure 10. After 5 seconds (b), a schematic has been produced in which map edges are close to meeting the Orientation constraint, while at 20 seconds (c) this constraint has been met more or less completely. There are few obvious improvements to the schematic after 100 seconds (d) – the reported reduction in cost is due to the optimization of the less important constraints (such as Length and Clearance). The results suggest that reasonably good schematics can be produced relatively quickly.

Figure 9 Relationship between schematic map cost and execution time (seconds). Data is averaged from 10 executions
How best to achieve a compromise between the quality of result and the time taken to produce the schematic would be dependant on the purpose.

4.5 Constraint Weighting

It is important to make sure that constraint costs are weighted appropriately; it is the weightings that govern the likelihood of any given generated map state being accepted during the annealing process. As such, the cost weightings must be set so as to accommodate any orders of precedence that might exist between the various constraint types. The schematics shown in Figure 11 were each generated using a different set of weightings. Figure 11(a) illustrates the problem of only considering Topological and Orientation constraints – the result is a map in which some edges appear to overlap, some edges are hardly visible and connectivity is difficult to discern. This situation improves slightly in Figure 11(b) where the Length constraint is also used. Figure 11(c) makes use of all implemented constraints and the result is a good schematic. In Figure 11(d) the required minimum edge length is given a relatively high value (100) – this gives rise to a situations in which other constraints cannot be met adequately (due to lack of free map space); the resulting schematic is of poor quality.

4.6 Example Application: Water Pipeline Network

Water pipeline networks are good examples of a practical application of schematic maps. They can provide a useful visualization tool for water engineers to help understand and
analyze the hydraulic conditions of a network. Figure 12 shows an example primary mains network of individual pipeline segments. Though connectivity details are explicitly recorded in the attribute table associated with the data, it is difficult to visualize the true connectivity in geographic terms as there are instances where the water pipelines lie very close to one other, but are not physically connected. Running the source pipeline data through the simulated annealing software produces a schematic map in which connectivity detail is clearly visible (due to the Clearance constraint).

5 Conclusions

This paper has focused on the development of automated means of generating schematic maps from large scale digital geographic datasets that are tailored for mobile GIS applications. Its key theme has been to describe and demonstrate the practical application of a new algorithm, which is based on the simulated annealing optimization technique. Prototype software has been produced and experimental results have shown the algorithm...
to be successful in producing schematic maps that meet user-defined constraints within reasonable time. The results presented also provide insight in how best to set the various input parameters required by the process, including the Douglas-Peuker tolerance value, process termination criteria and constraint cost weightings.

A possible shortcoming of the existing approach is that the annealing schedule has to be set manually; finding an appropriate set of parameter values can be time-consuming. The next stage of the research will investigate ways of automating this task. Zoraster (1997) suggests a method for generating the value $\tau$ (initial temperature) and this will provide a starting point.

Further work will concentrate on refining the technique through the use of additional constraints and additional feature classes. For example, the authors are working currently on allowing for the inclusion of POI features; the issue here will be to ensure that spatial relationships between POIs and network features are maintained subsequent to schematization. We also intend to do further work on the automated application of the appropriate small display specific symbology for the generated schematic map based on the referenced display scale to enhance visualization and usability. The intention is to build upon the work of Ware et al. (2003), which used simulated annealing as a means of controlling the feature displacements required to resolve graphic conflict resulting from the symbolization of road features. Another area for future work is the implementation and evaluation of alternative optimization techniques. Those earmarked for evaluation include Tabu Search and Genetic Algorithms.

References


© 2006 The Authors
Journal compilation © 2006 Blackwell Publishing Ltd
Douglas D H and Peuker T K 1973 Algorithms for the reduction of the number of points required to represent a digitized line or its character. Canadian Cartographer 10: 112–22
Saalfeld A 1999 Topologically consistent line simplification with the Douglas-Peucker algorithm. Cartography and Geographic Information Science 26: 7–18
Waldorf S P 1979 Schematic Navigational Map Design. Unpublished M.S. Thesis, Department of Geography, University of Kansas

Appendix: OffsetPolyline Function

Sub OffsetPolyline(n As Integer, xin() As Double, yin() As Double, closed As Boolean, d As Double, xout() As Double, yout() As Double)
Dim e, f, e1, e2, veck As vector3
Dim l As Double
Dim ist, ifin, i1, i As Integer
veck.x = 0
veck.y = 0
veck.z = 1
If (closed) Then
Else

\[ e_{2x} = x_{in}(0) - x_{in}(n-1) \]
\[ e_{2y} = y_{in}(0) - y_{in}(n-1) \]
\[ e_{2z} = 0 \]

If (ist <= ifin) Then

For i = ist To ifin

\[ e_{1x} = e_{2x} \]
\[ e_{1y} = e_{2y} \]
\[ e_{1z} = e_{2z} \]
\[ i1 = i + 1 \]

If (i1 = n) Then

\[ i1 = 0 \]

End If

\[ e_{2x} = x_{in}(i1) - x_{in}(i) \]
\[ e_{2y} = y_{in}(i1) - y_{in}(i) \]
\[ e_{2z} = 0 \]

Call normalise(e1)

Call normalise(e2)

Call cross(e1, veck, f)

\[ l = d \times (e_{1x} \times e_{2y} - e_{1y} \times e_{2x}) / (1 + dot(e1, e2)) \]

\[ x_{out}(i) = x_{in}(i) + d \times f.x + l \times e_{1x} \]
\[ y_{out}(i) = y_{in}(i) + d \times f.y + l \times e_{1y} \]

Next i

End If

End Sub

Function dot(a As vector3, b As vector3) As Double

\[ \text{dot} = a.x \times b.x + a.y \times b.y + a.z \times b.z \]
End Function

Sub normalise(a As vector3)
Dim lng As Double
lng = Sqr(dot(a, a))
If (lng > 0) Then
    a.x = a.x / lng
    a.y = a.y / lng
    a.z = a.z / lng
End If
End Sub

Sub cross(a As vector3, b As vector3, c As vector3)
c.x = a.y * b.z - a.z * b.y
    c.y = a.z * b.x - a.x * b.z
    c.z = a.x * b.y - a.y * b.x
End Sub