Ontology-driven map generalization

Lars Kulik*, Matt Duckham†, Max Egenhofer‡

* *Department of Computer Science and Software Engineering*
University of Melbourne, Victoria 3010, Australia

† *Department of Geomatics*
University of Melbourne, Victoria 3010, Australia

‡ *National Center for Geographic Information and Analysis, Department of Spatial Information Science and Engineering and Department of Computer Science*
348 Boardman Hall, University of Maine, Orono, ME 04469-5711, USA

Abstract

Different users of geospatial information have different requirements of that information. Matching information to users' requirements demands an understanding of the ontological aspects of geospatial data. In this paper, we present an ontology-driven map generalization algorithm, called DMin, that can be tailored to particular users and users' tasks. The level of detail in a generated map is automatically adapted by DMin according to the semantics of the features represented. The DMin algorithm is based on a weighting function that has two components: (1) a geometric component that differs from previous approaches to map generalization in that no fixed threshold values are needed to parameterize the generalization process and (2) a semantic component that considers the relevance of map features to the user. The flexibility of DMin is demonstrated using the example of a transportation network.

Key words:
"cartographic generalization," "line simplification," "geospatial information semantics," "task-oriented"

1 Introduction

Different users of geospatial information have different requirements of that information. For example, the information requirements of a tourist exploring the downtown area of a historic city are expected to be manifestly different from the requirements of a dispatch driver delivering goods to local businesses in the same city. The tourist is typically interested in locating landmarks and places of cultural and historical relevance, including scenic parts of the city; the delivery driver is

interested in the shortest path, avoiding heavy traffic and roadworks, and directly accessing the delivery addresses. In order to meet the particular requirements of a user engaged in a specific task, information retrieval and processing operations must be able to incorporate ontological information about the data, the user, and the user's task.

In this paper, we examine the inclusion of such ontological information within one class of geospatial information processing operation: generalization. Generalization concerns the process of producing maps at coarser levels of detail, while retaining essential characteristics of underlying geographic information (Weibel, 1995).

The growth of mobile and location-aware systems presents a new set of challenges for map generalization techniques. Traditional generalization techniques produce maps that are general-purpose, that is, applicable to a wide range of different tasks. Mobile users of location-aware systems require information that is directly relevant to the specific task in which they are engaged. At the same time, the limited communication, power, processing, and display characteristics of most mobile computing devices place constraints on the digital characteristics of information provided to users, in particular bandwidth constraints.

As a consequence, task-oriented generalization algorithms are needed that can adapt information to the diverse requirements of users of mobile and location-aware systems. In this paper we present an ontology-driven map generalization algorithm, called DMin ("dee-min"), that is able to meet this need ("DMin" is a contraction of "decimation-min-ε," see sections 2.1 and 3.1). DMin comprises two components: a geometric component and a semantic component. Following the review of background literature in section 2, we present DMin and explore its properties in section 3. Section 4 illustrates the geometric component of DMin, using the example of a transportation network. Section 5 shows, using the same transportation example, how the semantic component of DMin can be used to adapt the generalization process to a range of different task-oriented user requirements. Finally, the discussion in section 6 concludes the paper and presents suggestions for further work.

2 Background

Techniques for generalizing two-dimensional data in geospatial applications have a long tradition. Jones (1997) distinguishes eight different generalization techniques: elimination, simplification (also called reduction), typification, exaggeration, enhancement, collapse, amalgamation, and displacement. We confine our survey to line simplification techniques. In section 5 we also discuss the elimination of lines, which in turn can form the basis of amalgamation and collapse operations.

2.1 Line simplification algorithms

Heckbert and Garland (1997) give an extensive survey of polygonal simplification algorithms for lines as well as surfaces. These techniques may be thought of as lossy compression techniques. Simplified representations, such as a schematic map, conserve bandwidth when transmitted to a mobile computing device, for example a handheld or wearable computer. Typical two-dimensional visualizations of geospatial data include maps of transportation networks, for example street networks and terrain elevations. In such maps, (polygonal) lines are used to represent the extents of streets or differences in elevation.

Computational cartography has led to the development of a suite of techniques for simplifying and reducing detail in polygonal lines (McMaster, 1987; McMaster and Shea, 1992). Most of these simplification techniques employ a *decimation* strategy, in which the vertices of polygonal lines are deleted sequentially in accordance with a preset error criterion. An example of a simple decimation algorithm is to remove every n^{th} point not fulfilling an error criterion. More sophisticated decimation algorithms determine which points are to be deleted by superimposing local tolerance bands upon the original line (Reumann and Witkam, 1973). Other techniques process local direction and distance information (Jenks, 1981; McMaster, 1987), taking into account two or three points of a line and traversing the line sequentially, deleting vertices according to a predefined minimum length, or angle, or both.

One of the most popular techniques, also employed in commercial systems, is the Douglas-Peucker algorithm (Douglas and Peucker, 1973). In contrast to decimation algorithms, the Douglas-Peucker algorithm employs a constructive *refinement* strategy. Vertices are sequentially inserted between the extreme points of the original polygonal line in accordance with a preset error criterion. Algorithms like the Douglas-Peucker algorithm are often described as *global*, because they process an entire line at once. However, like most of these so-called global algorithms the Douglas-Peucker algorithm is only able to operate *globally* upon one polygonal line at a time. An important feature of the DMin algorithm is that it is able to operate upon multiple polygonal curves at the same time, therefore, optimizing *universally* across an entire data set. An alternative to this approach is to integrate global generalization algorithms into a universal generalization solution, for example using multi-agent systems (Lamy et al., 1999; Galanda and Weibel, 2002). However, the multi-agent approach has been criticized for requiring impractical levels of computational resources (Vermeij et al., 2003). Consequently, it is not well-suited to on-demand mapping in a mobile computing environment.

Every line simplification algorithm introduces a deviation from the original lines, the error, ε, of the simplification (Cromley, 1991). There are two types of design goals for simplifications algorithms with respect to error bounds (Imai and Iri, 1988):

(1) minimizing the number of vertices for a given error bound ε, called min-# problem; and
(2) minimizing the approximation error for a certain number of vertices, called min-ε problem.

According to Imai and Iri (1988) min-# problem are easier to solve than min-ε problems.

Most line simplification algorithms are min-# techniques and work with a predetermined error bound ε. Typically, the error bound defines a tolerance band such that only points within the tolerance band are allowed to be deleted. The disadvantage of this technique is that a user has to specify in advance a sensible threshold value of the error bound ε for each dataset. This threshold value indirectly determines the number of points removed. The actual number of points that will be removed is not known in advance, making min-# techniques less suitable for compression purposes.

2.2 Goals of line simplification

Weibel (1996) specifies four criteria that generalization techniques have to satisfy: Gestalt (shape) constraints, semantic constraints, metric constraints, and topological constraints. We consider each of these criteria in turn.

First, with notable exceptions like the Douglas-Peucker algorithm, many algorithms are not able to preserve the overall shape of lines, because the simplification criteria involve only *local* curve features. The shape of a curve is essential for identifying the salient features of the object represented by the curve. For example, a meandering line representing a street might indicate to a driver slow or otherwise difficult driving conditions (Mark, 1989). Recent work on line simplification in image processing (Latecki and Lakämper, 1999, 2000, 2002) has drawn upon psychological research into shape interpretation and human cognition (Hoffman and Singh, 1997).

Second, the majority of cartographic algorithms are not tailored to meet the requirements of a specific user or task. The algorithms are based primarily on geometric error criteria, but do not include semantic or ontological knowledge about lines, such as the meaning of a line (e.g., road versus vegetation boundary) or its non-spatial properties (e.g., surface type or traffic throughput of a road). Although the inclusion of semantic knowledge is widely regarded as an important issue for line generalization algorithms (Jones, 1997), current algorithms have not addressed this need. To date, only a few generalization algorithms, such as the multicriteria line generalization of Sinha and Flewelling (2002), are able to incorporate any non-spatial information into the generalization process.

Third, many line generalization algorithms are not guaranteed to be *optimal* ac-

cording to a given error criterion. In this context, "optimal" means an algorithm is able to find the best possible approximation that minimizes the overall deviation of the simplified line with respect to the original line. According to Heckbert and Garland (1997) there has been relatively little work on optimal simplification algorithms for lines. Imai and Iri (1988) have shown that the computational complexity of optimal decimation techniques for a curve with n vertices in general is $n^2 \log n$. Because optimal algorithms can be slow, it is common practice to trade optimality for speed. The Douglas-Peucker algorithm is not optimal, but can be computed with a complexity of $n \log^* n$ (Hershberger and Snoeyink, 1998).

Fourth, not all algorithms are guaranteed to be topologically consistent (Muller, 1990), and may introduce such topological inconsistencies as new self-intersections or new intersections with neighboring map features. The Douglas-Peucker algorithm is not necessarily topologically consistent. For map data in vector format, the importance of topological consistency has been emphasized by Bertolotto and Egenhofer (2001). Approaches simplifying lines in a topologically consistent manner are given in Jones et al. (1995), de Berg et al. (1998), and van der Poorten et al. (2002). Topologically consistent simplification algorithms are typically based on computational geometry techniques such as Delaunay triangulation (van der Poorten et al., 2002).

In summary, none of the current generalization algorithms is at the same time ontology-driven, shape-preserving, topologically consistent, and optimal.

3 The DMin algorithm

The DMin algorithm incorporates both geometric and semantic information about a line. The core aim of DMin is to achieve ontology-driven simplification, where semantic information concerning the relative importance of features to a user can be accounted for within the simplification process. Additional design goals of the DMin algorithm include shape-preservation and the maintenance of topological consistency. Although the DMin algorithm is not optimal, it is still very efficient in terms of computational complexity and, therefore, scales well to very large datasets.

3.1 General approach

A crucial foundation for DMin is the assignment of separate weights to each vertex of every line in a dataset. The weight consists of two components, a geometric and a semantic component. The DMin algorithm uses the weighting function to decide which points are deleted when the set of lines is simplified. At each step the algorithm determines for all lines the point with the smallest weight. Those vertices

with the smallest weights are successively dropped.

The DMin algorithm is a min-ε technique. At each iteration DMin removes the point with the least importance (weight) and consequently does not require any preset error bound. The total number of points to be deleted can be explicitly specified by the user. For example, if a user requires a compression ratio of 10:1, then the algorithm can be set to remove those 90% of points that are of least importance, in terms of both geometry and semantics. In the context of mobile and location-aware computing, the min-ε approach, used in DMin, enables the level of compression to be adjusted to the limitations of bandwidth or display resolution.

3.2 Weighting functions

The geometric component of DMin measures the overall impact of a vertex on the shape of the line to which it belongs. The geometric weight assigned to each vertex will usually depend on the length of the segments meeting at the vertex and their turning angle. In this case, the weighting function will be a ternary function $f : \mathbb{R}^3 \rightarrow \mathbb{R}$. The precise nature of the function may vary. It could depend linearly on the segments and the turning angle, that is, $f(s_1, s_2, \alpha) \mapsto s_1 \cdot s_2 \cdot \alpha$. Alternatively, it could emphasize the turning angle, or minimize the area of the triangle spanned by the two segments.

A key element of the DMin algorithm is its *adaptability*. DMin can be tailored to different cartographic or psychological requirements by adopting different weighting functions. In the examples that follow in later sections, we have primarily adopted the L^2 error norm to ensure compatibility with existing research. The L^2 error norm is a standard construction used in geometry to quantify the discrepancy between two curves, in terms of the area enclosed by those curves. Although the L^2 error norm is widely used in simplification algorithms, results of cognitive psychology and gestalt theory (Hoffman and Singh, 1997) show that salient angles are often crucial for a description of shape features. Thus, we have also included in section 4 a weighting function that emphasizes angular information.

In addition to the geometric impact of a vertex upon the entire polygonal curve, we also assign a weight for the semantic relevance of each vertex within a particular context. A geographic object is described not only by its geometric extension, but also by its semantic features. For example, in the case of a road network, an underlying road ontology might provide the semantics for different classes of roads. In turn, different road classes may assume varying levels of importance to a person engaged in a particular task (e.g., highways being more important than major roads, which are more important than minor roads). The semantic weight assigned to each vertex should reflect the relevance of that vertex to a specific task- or user-oriented context. Later sections give examples of such semantic weighting functions.

3.3 Topological consistency

The algorithms presented in sections 3.4 and 3.5 are topologically consistent: they neither introduce self-intersections for a single line nor do they introduce any new intersections with neighboring lines. Self-intersections and new intersections introduced during simplification can be identified by applying the triangle-criterion, which is used, for example, in knot theory (Livingston, 1993). Deleting a vertex from a polygonal curve introduces a self-intersection if and only if there exists one or more points of the curve lying within the triangle spanned by the two segments of the vertex in question (Figure 1).

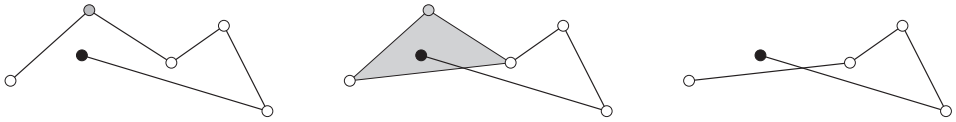

Fig. 1. Introducing intersections by deleting vertices

The triangle condition ensures that the line simplification does not introduce any new line intersections. In the case of non-planar graphs, however, the triangle condition is insufficient to guarantee that existing line intersections are preserved (Figure 2). The configuration shown in Figure 2 is not uncommon in geospatial information, for example in the representation of bridges and tunnels in transport networks. Removing these intersections would change the topological structure of the transport network. To ensure topological consistency the DMin algorithm maintains a list of line intersections that are not vertices of polygonal curves.

Fig. 2. Removing intersections by deleting vertices

Finally, our topological consistency checks assume that the underlying graph is connected. For a disconnected graph there is a possibility that the DMin algorithm could change the graph's topology, if two of the graph's disconnected non-intersecting components still have no intersection after one component is simplified (Figure 3). Although such topological inconsistencies are *possible* for disconnected graphs, they represent a degenerate case and seem unlikely to occur for spatial data.

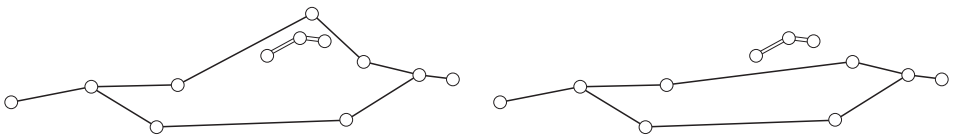

Fig. 3. Line simplification leading to topological inconsistency in disconnected graphs

3.4 Line simplification algorithm

The first variant of the DMin algorithm, called S-DMin, *simplifies* the polygonal curves up to a desired compression level; that is, until a certain number of vertices have been removed. Every vertex of a polygonal curve that is not an endpoint is called an *interior* vertex. The S-DMin line simplification algorithm only deletes interior vertices of polygonal curves. If N is the overall number of interior vertices of all polygonal curves and n the number of removed interior vertices, then the compression level is the ratio n/N expressed as a percentage, where $N > 0$. A compression level of 100% (assuming this is possible without introducing self-intersections) means that the simplification algorithm removes every interior point of each polygonal curve, resulting in a planar straight line graph.

Algorithm 1: S-DMin semantic line simplification

Input: A set P of polygonal curves and the desired compression level l

Output: A set P of polygonal curves simplified to the desired compression level l

// Initialize weights;
foreach *curve* $c \in P$ **do**
    foreach *interior vertex* v *of* c **do**
        Compute geometric weight $\omega_g(v)$;
        Compute semantic weight $\omega_s(v)$;
        Compute combined weight $\omega(v) \leftarrow \omega_g(v) \otimes \omega_s(v)$;
Generate a priority queue Ω of all weights ω;
1.1 Compute a list I of intersection points that are not vertices of a curve;
// Iterate over weights;
Set $n \leftarrow 1$;
while *current compression level* $< l$ *and* $\Omega \neq \emptyset$ **do**
    Select vertex v_i with n^{th} smallest weight from Ω;
    Select curve $c \in P$ containing v_i;
1.2 $\Lambda(v_i) \leftarrow \{v | v \neq v_i$ and v is a c-adjacent vertex$\}$;
1.3 **if** $\overline{v_{i-1}v_i} \cap I = \emptyset$ *and* $\overline{v_iv_{i+1}} \cap I = \emptyset$ *and* $\Lambda(v_i) \cap \Delta(v_{i-1}v_iv_{i+1}) = \emptyset$ **then**
        Remove vertex v_i from c;
        Remove $\omega(v_i)$ from Ω;
        Set $n \leftarrow 0$;
    Set $n \leftarrow n + 1$;

Algorithm 1 initializes the weight for every interior vertex v as a combination of the geometric and semantic weights for that vertex, $\omega_g(v)$ and $\omega_s(v)$ respectively. Each weight is stored in a priority queue Ω. At line 1.1, a list I of existing intersection points is stored. These points can be junctions in the road network or represent tunnels and bridges (see section 3.3).

At each iteration the algorithm tries to remove the vertex with the minimum weight.

When deleting a vertex of a curve, the algorithm must check for intersections with other curves. To increase efficiency, instead of checking for intersections of the curve c with all other curves, the algorithm only needs to check for intersections with those curves that are in some sense in the immediate vicinity of c. For example, for a planar graph it would only be necessary to check for intersections with those parts of curves that share a common face with c. In the more general case of non-planar graphs, we require a more complex definition of what it means to be in the "immediate vicinity" of c, leading to the definitions below.

A polygonal region is called *simple* if it has a non-intersecting boundary consisting of edges and nodes in the underlying graph. A simple polygonal region is a c*-face* is if has at least one line segment in common with the curve c. A c-face is *minimal* if it does not completely contain any other c-face. A boundary vertex of a minimal c-face is called c*-adjacent*. The algorithm maintains a set Λ of all vertices that are c-adjacent, in order to check more efficiently for intersections.

The line segment connecting two vertices v_i and v_{i+1} is denoted by $\overline{v_i v_{i+1}}$. The condition in line 1.3 checks whether removing the selected vertex v_i either:

(1) deletes an existing interior point of I (i.e., one of segments with endpoint v_i contains a point of I); or
(2) introduces an intersection (i.e., one of the vertices of Λ lies in the triangle defined by the segments with endpoint v_i).

If the elimination of the vertex does not change the topology of the graph, the vertex is removed from its supporting curve and its weight deleted from Ω. Otherwise, the algorithm removes the vertex with the next-smallest weight that does not introduce an intersection. The algorithm terminates once the desired compression level is reached, or if there exist no curves that can be simplified without changing the graph's topology.

3.5 Line generalization algorithm

The S-DMin algorithm simplifies but never eliminates curves. However, we can use the general form of the line simplification algorithm to define a line generalization algorithm that may eliminate complete curves from the graph (Algorithm 2). The second algorithm, called SE-DMin for "simplification-elimination-DMin," assigns a weight to every vertex, including the endpoints of each curve (line 2.1). The algorithm assumes that the weights of the endpoints of a curve are always greater than any of the weights of the interior points of a curve. This implies that the line generalization process will always delete interior points of a curve before deleting its endpoints. The intuition behind the SE-DMin algorithm is that a line is first simplified before it is completely deleted.

Algorithm 2: SE-DMin semantic line generalization

Input: A set P of polygonal curves and the desired compression level l

Output: A set P of polygonal curves generalized to the desired compression level l

```
// Initialize weights;
foreach curve c ∈ P do
    foreach interior vertex v of c do
        Compute geometric weight ω_g(v);
        Compute semantic weight ω_s(v);
2.1     Compute combined weight ω(v) ← ω_g(v) ⊗ ω_s(v);
Generate a priority queue Ω of all weights ω;
// Iterate over weights;
Set n ← 1;
while current compression level < l and Ω ≠ ∅ do
    Select vertex v_i with n^th smallest weight from Ω;
    Select curve c ∈ P containing v_i;
    if no intersections are introduced then
        if v_i is an interior vertex of c then
            Remove vertex v_i from c;
            Remove ω(v_i) from Ω;
        else
            Select endpoint v_j ≠ v_i of c;
2.2         Remove ω(v_i) and ω(v_j) from Ω;
            Remove c from P;
        Set n ← 0;
    Set n ← n + 1;
```

The initialization stage of SE-DMin assigns to every vertex of each curve a weight. The algorithm continues to delete points with minimal weights until the desired compression level is reached or the graph cannot be further generalized without introducing new intersections between curves. If a vertex has a minimal weight and its deletion does introduce any intersections between curves, two cases can occur:

(1) the vertex is an inner point of a curve; or
(2) the vertex is an endpoint of a curve.

In case 1 only the point is deleted from its corresponding curve and its weight is deleted from the list Ω. In case 2, however, the entire curve to which the vertex belongs is eliminated. At line 2.2 the weights of both endpoints belonging to this curve are deleted from the list Ω. In contrast to the simplification algorithm, the generalization algorithm can change the topology of the graph by removing lines. Existing intersections between lines may be removed. Therefore, the condition in S-DMin algorithm 1 (line 1.3) that checks whether an intersection is removed is

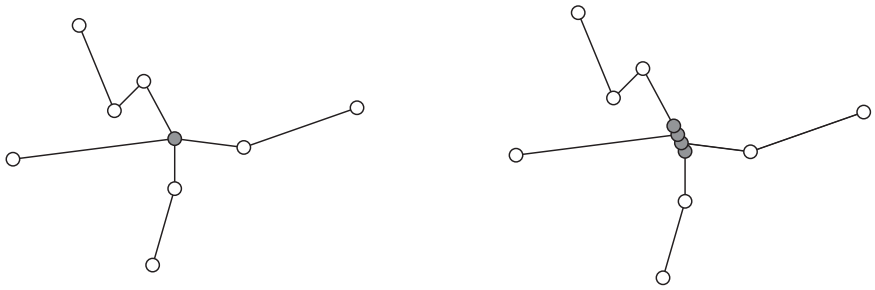

Fig. 4. Four co-located endpoints at an intersection

missing in SE-DMin algorithm 2.

In order to delete an entire curve from a graph, we assume that an intersection of curves is represented not by a single point, but by the unique points that belong to each individual curve meeting at that intersection. For example, in Figure 4 even though the points are coincident in the representation, each point can be still identified with the curve to which it belongs. The number of coincident points at an intersection corresponds to the number of curves incident with that intersection (the degree of that intersection). For example, if four streets are incident at an intersection, the intersection is represented by four endpoints. Given a four-way intersection, the deletion of one street would, therefore, result in the removal of the one curve, but not the intersection point itself, resulting in a T-junction.

An optimal simplification algorithm guarantees that there is no better approximation of a line simplification with respect to a given weighting function. Because the primary goal of this approach is to explore the value of combining semantic with geometric information for generalization, we implemented a fast algorithm and traded optimality for speed. The DMin algorithm is locally optimal, that is, optimal in each step, but not globally optimal. However, any decimation technique can easily be modified to produce an optimal algorithm, albeit at the cost of greatly increased computational complexity (see Imai and Iri, 1988, for more details).

4 Example geometric simplification

This section illustrates the basic geometric simplification that can be performed by the S-DMin simplification algorithm. The algorithm was implemented, using Java, and tested on a range of real and simulated data. The examples in this and following sections are based on topographic data for British Columbia, Canada [*] .

[*] The datasets used in this paper are available via the Canadian Geospatial Data Infrastructure (CGDI), `http://www.cgdi.gc.ca`. Data for Figures 5, 6, 8–11 © 2003 Government of Canada with permission from Natural Resources Canada.

A design feature of the DMin algorithm is that it possesses sufficient adaptability to generalize using a wide variety of weighting functions (section 3.2). This feature contrasts with much of the previous work on generalization (section 2), which is concerned with the properties of specific weighting functions. To illustrate this property, Figure 5 shows the results of simplifying 90% of the points within a single line (part of a coastline) using three different geometric weighting functions.

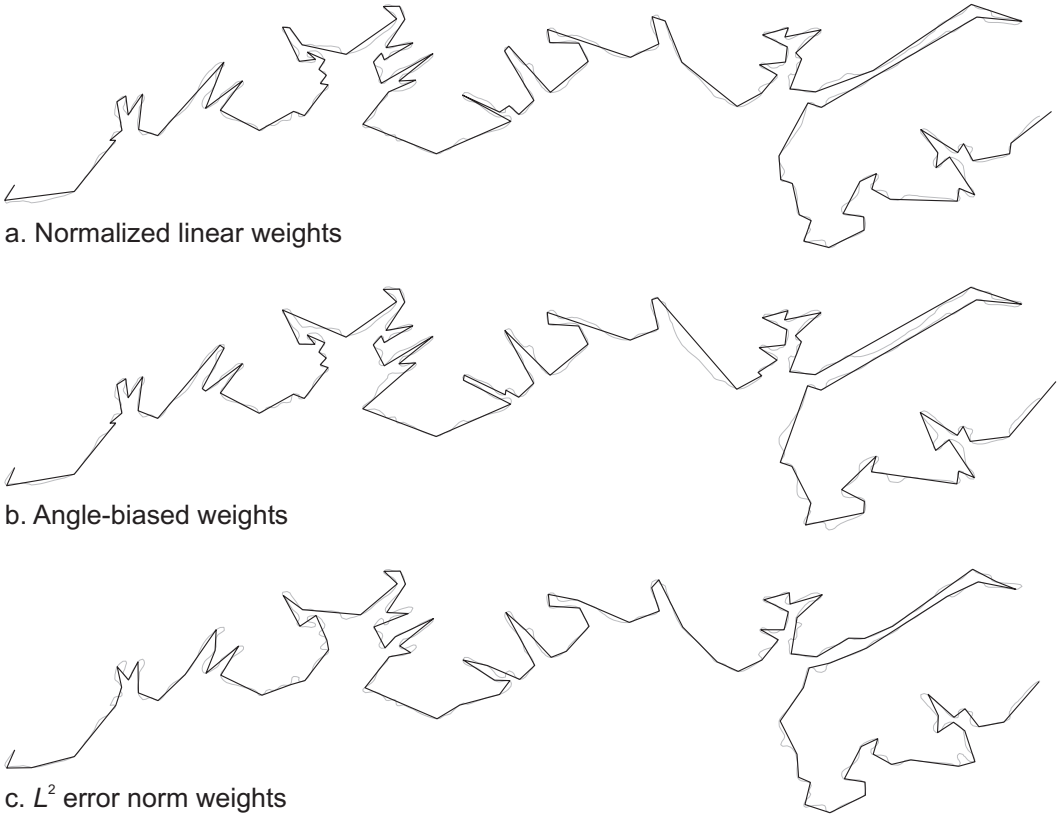

Fig. 5. Comparison of three different geometric line simplification weighting functions for part of the coastline of British Columbia, Canada with 90% of vertices removed.

The three geometric weighting functions used to generate Figure 5 are as follows:

(1) *Normalized linear weights*: After Latecki and Lakämper (1999), we tested the ternary weighting function $f(s_1, s_2, \alpha) \mapsto \frac{s_1 \cdot s_2 \cdot \alpha}{s1+s2}$, where, for three adjacent points p_{n-1}, p_n, and p_{n+1}, s_1 is the distance between p_{n-1} and p_n, s_2 is the distance between p_{n+1} and p_n, and α is the turning angle $p_{n-1}p_np_{n+1}$.

(2) *Angular-biased weights*: In keeping with Hoffman and Singh (1997), the weighting function can be biased toward the turning angle by using the ternary weighting function $f(s_1, s_2, \alpha) \mapsto s_1 \cdot s_2 \cdot \alpha^3$, where s_1, s_2, and α are defined as above.

(3) *L^2 error norm weights*: A commonly used error criterion in generalization, the L^2 error norm weights associated with each vertex are computed as the area enclosed between the original curve and the curve that would result from removing that vertex.

These three functions were chosen as a representative sample of the geometric weighting functions tested during the course of this work. In computational terms, the normalized linear and angle-biased weighting functions are the simplest to compute as the weighting for each point depends only on the distances and angle between three points. Weighting function 3, the L^2 error norm, may require a consideration of all the points within a line in order to compute the weight for a particular point.

In cartographic terms, all three weighting functions are sensible, allowing a high degree of simplification while still retaining the essential characteristics of the curve. The choice of which weighting function to use within a particular application will be, at least partly, a subjective one. From the perspective of this paper, the important message is that the DMin algorithm is flexible enough to allow any of these functions to be used to build the overall geometric weighting function ω_g, and indeed any similar functions.

As discussed in section 2, the DMin algorithm may be applied universally to a complex dataset made up of multiple lines and polygons, as to a single line. Figure 6 shows an example simplification of part of the transport network of British Columbia. At each iteration, the simplification process finds the optimal point to remove from across *all* the geometries in the dataset. By contrast, traditional generalization algorithms operate locally on part of a feature or globally on just one feature at a time. Per-feature generalization can be iterated over the whole dataset. However, processing universally across an entire dataset ensures that the simplification process operates within the context of the dataset as a whole.

4.1 Analysis

Figure 7 compares the behavior of the three geometric weighting functions for the data set displayed in Figure 6. For each weighting function, the figure plots the level of simplification (as percentage of points removed) against the total inaccuracy in the map (in terms of the L^2 error norm, the area of discrepancy between the generalized and ungeneralized map in square kilometers). The figure shows that all weighting functions perform well, in the sense that between 30% and 40% of the data points can be removed using the S-DMin simplification algorithm with almost zero error. The level of inaccuracy remains relatively low, up until the 60–80% generalization level, after which levels of inaccuracy increase dramatically.

The errors for the L^2 norm weighting function are initially marginally lower than for the other two weighting functions. In the 60–90% range this situation is reversed, with errors for the L^2 norm weighting function being larger than for the other two functions. This feature highlights the local optimality of the DMin algorithm. At each iteration, the L^2 error norm weighting function removes that vertex

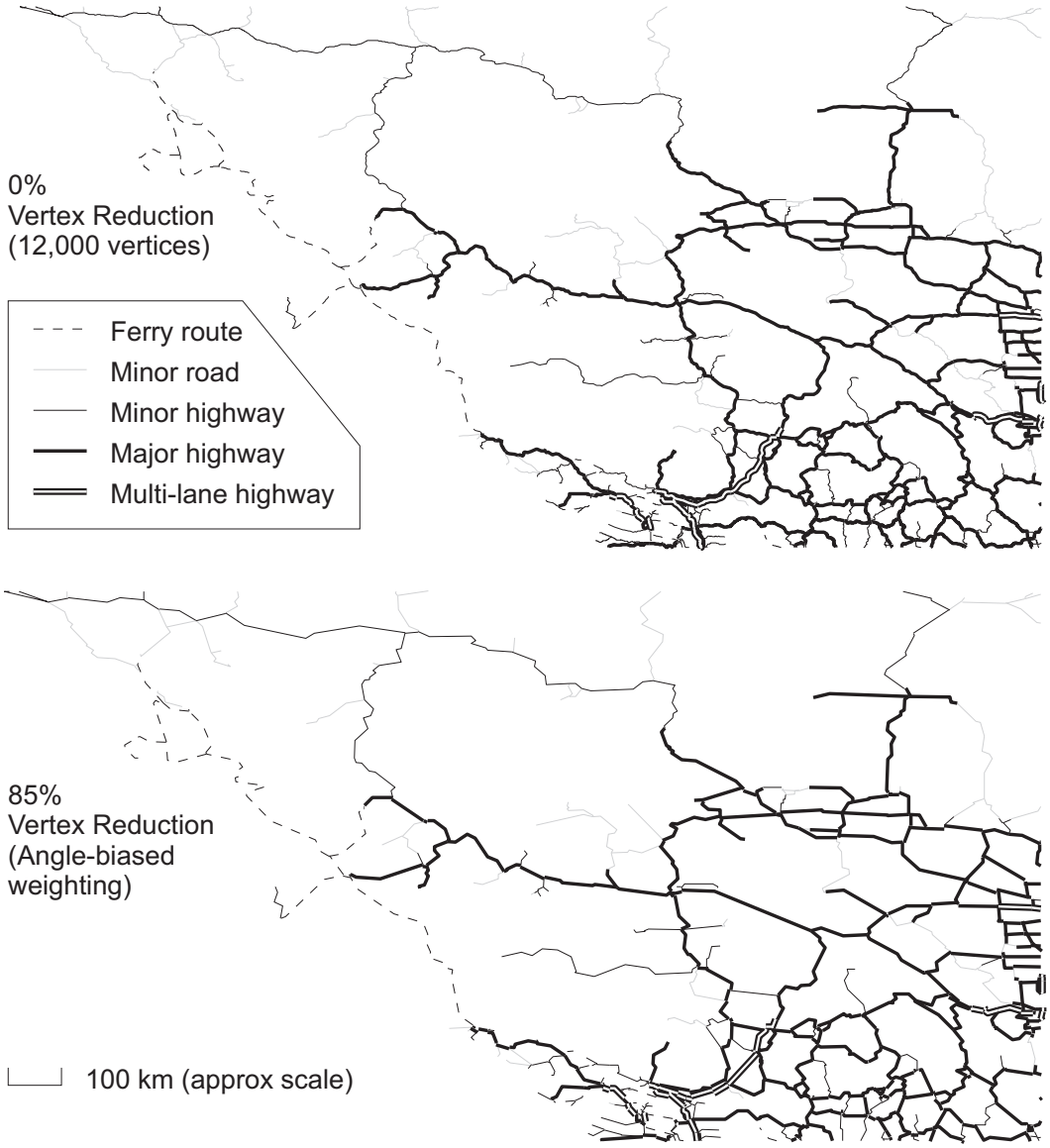

Fig. 6. Geometric line simplification for part of the transport network of British Columbia, Canada.

which yields the minimal increase in error. Over multiple iterations this strategy does not necessarily lead to a globally optimal solution (hence, the other weighting functions are able to achieve lower levels of inaccuracy).

4.2 Summary

This section has illustrated the geometric simplification of geospatial data using the DMin algorithm. DMin provides several significant advantages over many other conventional line simplification algorithms. Specifically, DMin is:

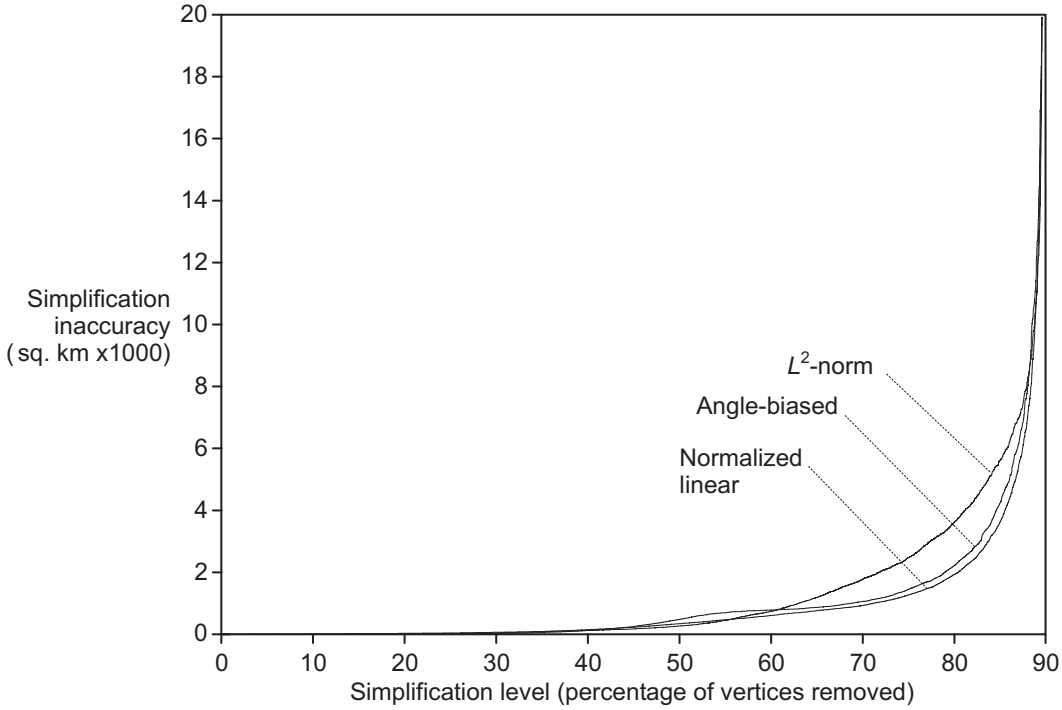

Fig. 7. Comparison of geometric line simplification weighting function performance for the transport network shown in Figure 6.

- able to generalize within the context of the entire dataset, not simply on a per-feature basis;
- flexible enough to support a range of weighting functions; and
- locally optimal (optimal at each iteration), although not globally optimal.

These features mean the DMin algorithm compares well with conventional line simplification algorithms. However, its advantage is that the weighting functions may be further extended to encompass semantic as well as geometric information, explored further in the following section.

5 Ontology-driven simplification

Traditional map generalization usually aims to provide a multi-purpose representation of a geographic environment, suitable for a variety of different users engaged in a diverse range of activities. The growth in mobile and location-aware computing in particular has led to a greater demand for mapping services that are tailored to the requirements of a particular user engaged in a specific task. In turn, this requires the ability to incorporate ontological information about the tasks and goals of a user when responding to user queries.

5.1 Parameterized semantic weighting function

The semantic weighting component of the DMin algorithm is just as flexible as the geometric weighting component. Potentially any semantic weighting function can be used. As an example, consider a weighting function based on a tuple of user-defined weights for each feature class in the map. The transportation dataset used to evaluate the algorithm has five feature classes. Correspondingly, the quintuple $\langle a, b, c, d, e\rangle$ can be used to encode the user-defined weights a, b, c, d, and e expressing the relevance to a user of ferry routes, minor roads, minor highways, major highways, and multi-lane highways, respectively. For the purposes of this example, we define a semantic weighting function (Equation 1), where k is some constant and x is the user-defined weight (from our tuple) for the feature class to which vertex v belongs:

$$\omega_s(v) \mapsto k^x \tag{1}$$

Using this approach, the ontology-driven simplification process can be parameterized using the tuple $\langle a, b, c, d, e\rangle$. The values of each element in the tuple can be manipulated to reflect a particular user's requirements. For example, the task ontology of a delivery driver, who considers multi-lane and major highways more relevant to the task of delivering goods than minor roads and highways or ferry routes, is modeled as the tuple $\langle 0, 0, 0, 1, 1\rangle$. By contrast, a tourist's information requirements might be represented as the tuple $\langle 1, 0, 1, 1, 0\rangle$ (i.e., tourists regard multi-lane highways and minor roads as of less relevance to touring than ferry routes, and minor and major highways). Similarly, a recreational cyclist who wishes to avoid heavy motorized traffic might regard minor roads and highways as of greatest relevance to the task of cycle-touring, represented as the tuple $\langle 0, 2, 1, 0, 0\rangle$.

To incorporate semantic information into the simplification process, we must combine the parameterized semantic weighting function with a geometric weighting function (section 3.4). For example, Figure 8 shows a detail from part of the transport network dataset, following simplification using the linear combination of geometric and semantic weighting functions, $\omega(v) \mapsto \omega_g(v) \cdot \omega_s(v)$. In Figure 8, ω_g is the L^2 error norm weighting and ω_s is the tuple-based semantic weighting function discussed above. The two maps represent the results of tailoring the simplification algorithm toward the task-oriented requirements of two different users, the delivery driver and the recreational cyclist.

Comparison of the maps in Figure 8 reveals that, while all the roads have been generalized to a high extent, greater detail does indeed remain on features of particular relevance to the user's task ontology. To quantify the differential generalization, Figure 9 shows an accuracy map for the entire dataset based on the delivery driver's task ontology, using tuple $\langle 0, 0, 0, 1, 1\rangle$. For each line segment, the level of accuracy of the simplified line has been computed as the L^2 error norm for that line segment, normalized according to the length of line segment. In Figure 9 black lines indicate zero error (i.e., no deviation from the dataset); dark gray lines show simplified line

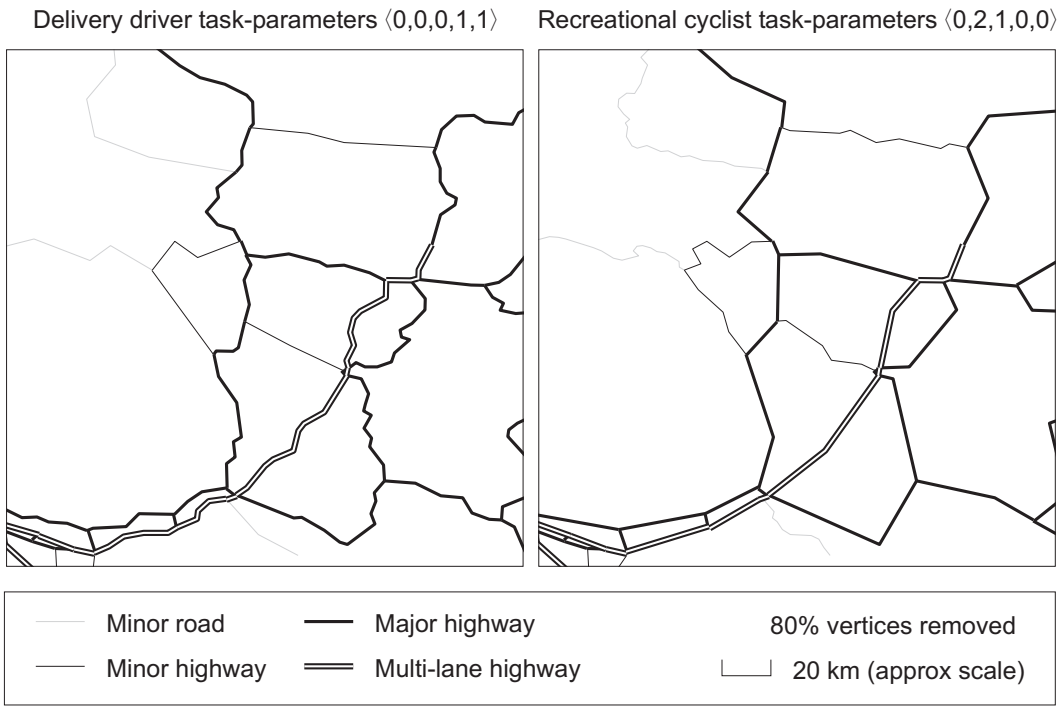

Fig. 8. Ontology-driven line simplifications using different task-oriented parameters.

segments of high accuracy; light gray lines show lower accuracy simplified line segments; and the palest gray shows those lines with the greatest normalized error levels. As expected, the spatial distribution of simplification accuracy indicates more accurate simplification is coincident with the features of interest, in this case major and multi-lane highways.

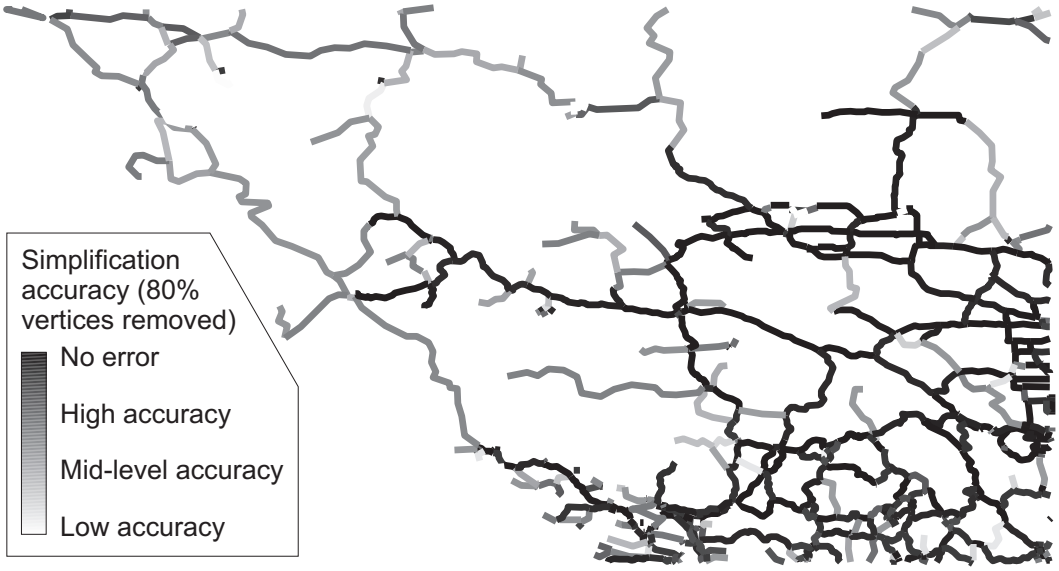

Fig. 9. Accuracy map for delivery driver task-oriented simplification.

Further statistical analysis, using a paired t-test for differences between two means, compared the performance of the ontology-driven simplification process (using the

delivery driver task-oriented parameters from Figure 8) with the standard geometric simplification (using the L^2 error norm). For each road class, the null hypothesis, that there exists no significant difference between the error characteristics of the two simplification techniques, was tested. After removing 80% of points in the dataset, the tests revealed significantly lower errors for multi-lane highways and major roads using the ontology-driven simplification process (significant at the 5% level). No significant differences between the error levels for the two simplification processes were discovered for any other road classes. This is an encouraging result as it indicates that:

(1) the ontology-driven generalization process leads to decreased error levels for the feature classes that are considered more relevant; and
(2) any corresponding loss of accuracy across less relevant feature classes is relatively mild (i.e., not statistically significant).

In these examples, the tuple weights needed to generate the task-oriented maps were the result of a subjective assessment of a potential users' needs. The prototype software developed for this research automatically includes a series of sliders as part of its graphical user interface, one slider for each class of transportation route. By adjusting the sliders, a user is able to interactively set the tuple-based semantic weights used in the generalization process.

A variety of different mechanisms might be used to set the semantic weights. For example, a generalization system could include a series of preset profiles, each associated with particular tuple. A user would then select the preset profile that best matches his or her own requirements. It is also conceivable that the semantic weights could be automatically generated by *user agents*, software programs that mediate on behalf of the user to acquire information that is more relevant to that user. The InfoSleuth architecture is one example of a system that utilizes such user agents in ontology-based information capture and retrieval (Nodine et al., 2000).

5.2 Semantic weighting with user routes

The parameterized semantic weighting function is just one possible way to incorporate task-oriented information about a user's information requirements into the simplification process. Figure 10 shows the accuracy map resulting from another type of task-oriented simplification. In Figure 10, the spatial distribution of simplification is controlled by proximity to a user's route through the transport network. Locations further away from the user's route are simplified in preference to those closer to the route. The semantic weights, ω_s, are derived from the familiar inverse distance decay function (Equation 2), where for some vertex v and user route r,

$d_r(v)$ is the minimum distance from v to some part of r:

$$\omega_s(v) = \frac{1}{d_r(v)} \tag{2}$$

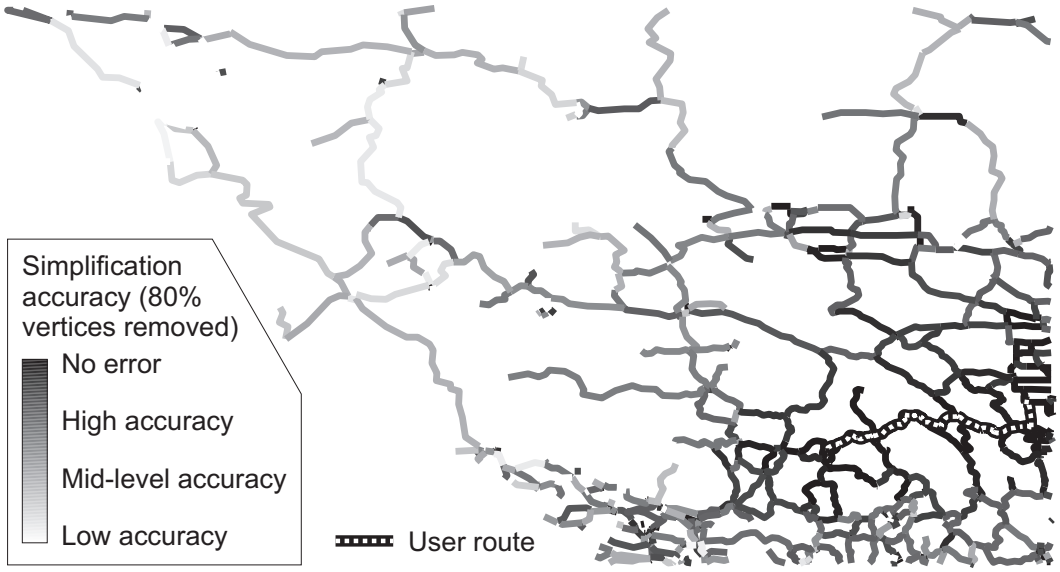

Fig. 10. Accuracy map for task-oriented simplification with user route.

This semantic weighting function could be used to provide a task-oriented generalization of a map for a user navigating through the transport network. Locations on or close to the user's route are clearly more relevant to the user's navigation task than those further away from the route. Correspondingly, these will be subject to lower levels of simplification.

5.3 Task-oriented elimination

The approach so far has focused exclusively on simplification. However, elimination may also be effected using the SE-DMin algorithm (section 3.5). In this section we briefly illustrate some results of using both simplification and elimination on the test network dataset. Figure 11 shows an extreme example of elimination in ontology-driven map generalization. The distance decay semantic weighting function (section 4), combined with the usual L^2 error norm geometric weighting function, has been used to generate Figure 11.

In Figure 11, the generalization level has been set to 90%. At this high level of generalization, the majority of roads have been eliminated, although the map still contains considerable detail close to the user's intended route. The SE-DMin algorithm as presented here is somewhat naive, causing a few roads at the periphery of the test area to become disconnected from the remaining transport network. A

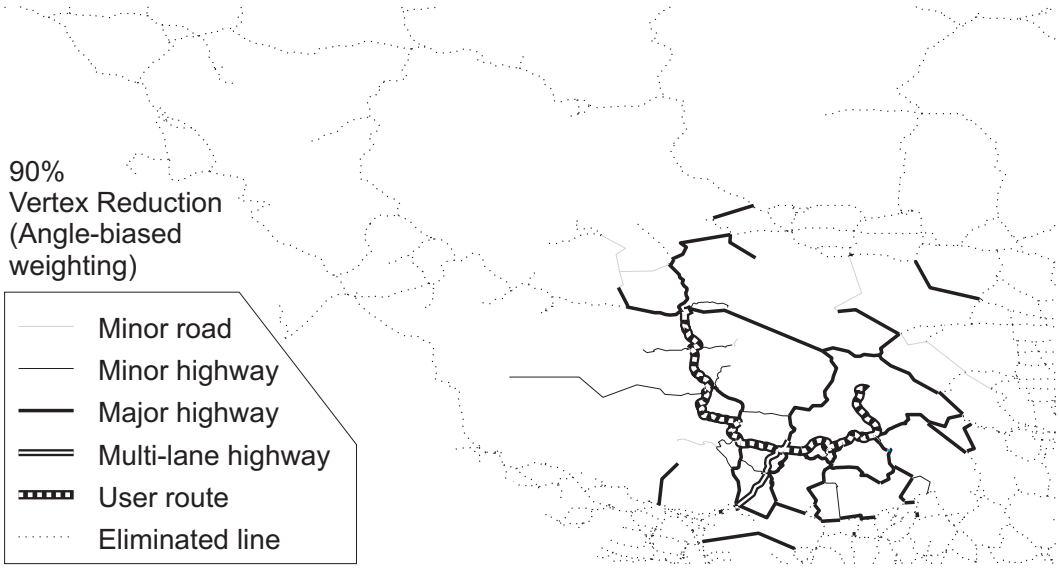

Fig. 11. Task-oriented elimination with user route.

more sophisticated weighting function could also incorporate topological weights that would prevent such disconnections.

5.4 Outlook

The examples in this section concern the task-oriented generalization of a road map based on an ontology for roads. Further refinements of this technique could also integrate ontological information concerning decision points and regions adjacent to roads. Decision points (primarily intersections in the road network) could be assessed with respect to their relevance to the navigation task. Decision points that are harder to negotiate or require a particular user action, such as a turn, could be presented at a higher level of detail than other intersections (cf. the incorporation of an ontology of turns into the wayfinding task in Duckham and Kulik, 2003). Figure 12 illustrates this concept using two different generalizations of a highway intersection. The left-hand diagram in Figure 12 shows the complete intersection. The other two diagrams in Figure 12 show generalizations of the complete intersection for a user who's task is to turn left onto another highway (central diagram) or to continue straight on (right-hand diagram).

Similarly, an underlying ontology of the regions that are adjacent to and enclosed by the roads might also be included (Fonseca and Egenhofer, 1999, have emphasized the importance of including such ontological information within GIS). Whether regions are aggregated by DMin, eliminating the boundary that separates two regions, would depend on their ontological similarity. Rodriguez and Egenhofer (2003) and Jones et al. (2003) discuss two examples of ontological similarity measures, which could be used as a semantic weighting component of the DMin algorithm. The

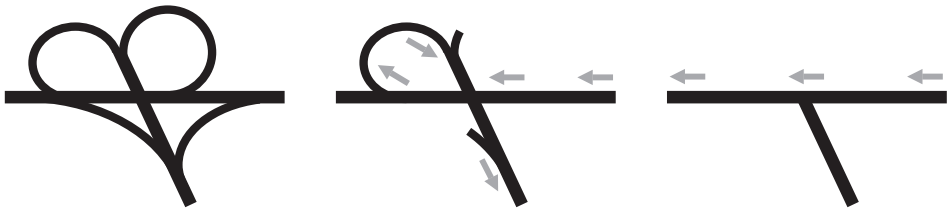

Fig. 12. A highway intersection (left) and two task-dependent generalizations of the intersection (center and right).

more similar two regions are, the more likely it would be that they would be amalgamated. In the case of the SE-DMin algorithm, the line adjacent to two similar regions could receive a low weighting, so ensuring that this line would be preferentially eliminated. These extensions, and many other ontology-driven generalization operations, could be achieved simply by varying the form of the semantic weighting function.

6 Discussion

This paper has described a new generalization algorithm that comprises both a geometric and a semantic component. The DMin algorithm is computationally efficient, operates universally across an entire dataset, and in the case of the S-DMin variant is topologically consistent. A wide range of shape preserving weighting functions can be used with the geometric component. The semantic component enables the generalization of geospatial data to be adapted to a specific user's requirements, preferentially generalizing those features that are of lower relevance or importance to a user. Experiments using DMin have demonstrated algorithm's adaptability to different types of user requirements, and the diversity of geometric and semantic weighting functions that can be integrated with the algorithm.

Future work in this area will need to address issues in at least three different areas arising from this initial research.

(1) *Integration with user agents*: A long-term goal of developing an ontology-driven generalization algorithm is to integrate the generalization process with autonomous user agents (section 5.1). These user agents aim to capture semantic information about the user and the user's tasks and goals, based on an analysis of the user's behavior patterns and context-aware sensor technology (such as location or motion sensors). User agents could then mediate on behalf of the user to automatically parameterize the generalization process to provide task-oriented geospatial information to users, without the need for any explicit user intervention in the process.
(2) *User testing*: The analysis of the ontology-driven generalization process presented in this paper has focused on the statistical characteristics of the gener-

alized information and the spatial distribution of errors within the generalized information. In combination with the development of user agents, user testing is needed to verify the suitability of generalized information for use with in specific application domains.

(3) *Generalization operators*: This paper has primarily concentrated on two generalization operations: line simplification, using the S-DMin algorithm, and elimination, achieved by the SE-DMin algorithm. Elimination itself can be seen as the initial stage of other generalization operations, such as collapse and amalgamation. Consequently, future work will concentrate on broadening the range of generalization operations that can be achieved within an ontology-driven framework.

Acknowledgements

This work was partially support by the National Geospatial-Intelligence Agency under grant numbers NMA201-00-1-2009, NMA201-01-1-2003, and NMA401-02-1-2009, and the National Science Foundation under grant numbers IIS-9970123 and EPS-9983432. Dr Duckham and Dr Kulik are partially supported by Early Career Researcher Grants from the University of Melbourne. Finally, the authors are grateful to the three anonymous reviewers for their constructive comments.

References

Bertolotto, M., Egenhofer, M. J., 2001. Progressive transmission of vector map data over the world wide web. GeoInformatica 5 (4), 345–373.

Cromley, R. G., 1991. Hierarchical methods of line simplification. Cartography and Geographic Information Systems 18 (2), 125–131.

de Berg, M. T., van Kreveld, M., Schirra, S., 1998. Topologically correct subdivision simplification using the bandwidth criterion. Cartography and Geographic Information Systems 25 (4), 243–257.

Douglas, D. H., Peucker, T. K., 1973. Algorithms for the reduction of the number of points required to represent a line or its caricature. The Canadian Cartographer 10 (2), 112–122.

Duckham, M., Kulik, L., 2003. "Simplest" paths: Automated route selection for navigation. In: Kuhn, W., Worboys, M. F., Timpf, S. (Eds.), Spatial Information Theory: Foundations of Geographic Information Science. Vol. 2825 of Lecture Notes in Computer Science. Springer, pp. 182–199.

Fonseca, F. T., Egenhofer, M. J., 1999. Ontology-driven geographic information systems. In: Medeiros, C. B. (Ed.), Proceedings Seventh Symposium on Advances in Geographic Information Systems. pp. 14–19.

Galanda, M., Weibel, R., 2002. An agent-based framework for polygonal subdivision generalisation. In: Advances in Spatial Data Handling. pp. 209–224.

Heckbert, P. S., Garland, M., 1997. Survey of polygonal surface simplification algorithms.

Hershberger, J., Snoeyink, J., 1998. Cartographic line simplification and polygon CSG formula in $O(n \log^* n)$ time. Computational Geometry 11 (3-4), 175–185.

Hoffman, D. D., Singh, M., 1997. Salience of visual parts. Cognition 63, 29–78.

Imai, H., Iri, M., 1988. Polygonal approximations of a curve – formulations and algorithms. In: Toussaint, G. T. (Ed.), Computational Morphology. North-Holland Publishing Company, Amsterdam, pp. 71–86.

Jenks, G., 1981. Lines, computers and human frailties. Annals of the Association of American Geographers 71 (1), 1–10.

Jones, C., 1997. Geographical Information Systems and Computer Cartography. Longman, Harlow.

Jones, C. B., Alani, H., Tudhope, D., 2003. Geographical terminology servers—closing the semantic divide. In: Duckham, M., Goodchild, M. F., Worboys, M. F. (Eds.), Foundations of Geographic Information Science. Taylor & Francis, London, pp. 205–222.

Jones, C. B., Bundy, G., Ware, J., 1995. Map generalisation with a triangulated data structure. Cartography and Geographic Information Systems 2 (4), 317–331.

Lamy, S., Raus, A., Demazeau, Y., Jackson, M., Mackaness, W., Weibel, R., 1999. The application of agents in automated map generalisation. In: Proc. 19th International Cartographic Conference. pp. 1225–1234.

Latecki, L. J., Lakämper, R., 1999. Polygon evolution by vertex deletion. In: Proceedings of the 2nd International Conference on Scale-Space Theories in Computer Vision. Springer, Corfu, Greece, pp. 398–409.

Latecki, L. J., Lakämper, R., 2000. Shape similarity measure based on correspondence of visual parts. IEEE Transaction Pattern Analysis and Machine Intelligence 22 (10), 1185–1190.

Latecki, L. J., Lakämper, R., 2002. Application of planar shape comparison to object retrieval in image databases. Pattern Recognition 35 (1), 15–29.

Livingston, C., 1993. Knot Theory. Vol. 24 of The carus mathematical monographs. Mathematical Association of America, Washington, DC.

Mark, D. M., 1989. Conceptual basis for geographic line generalization. In: Proceedings, Ninth International Symposium on Computer-Assisted Cartography (Auto-Carto 9), Baltimore, MD. pp. 68–77.

McMaster, R., Shea, K. S., 1992. Generalization in Digital Cartography. Association of American Geographers.

McMaster, R. B., 1987. Automated line generalization. Cartographica 24 (2), 74–111.

Muller, J.-C., 1990. The removal of spatial conflicts in line generalization. Cartography and Geographic Information Systems 17 (2), 141–149.

Nodine, M., Fowler, J., Ksiezyk, T., Perry, B., Taylor, M., Unruh, A., 2000. Active information gathering in InfoSleuth. International Journal of Cooperative Information Systems 9 (1/2), 3–28.

Reumann, K., Witkam, A., 1973. Optimizing curve segmentation in computer graphics. In: International Computing Symposium. North-Holland Publishing Company, pp. 467–472.

Rodriguez, M., Egenhofer, M., 2003. Determining semantic similarity among entity classes from different ontologies. IEEE Transactions on Knowledge and Data Engineering 15 (2), 442–456.

Sinha, G., Flewelling, D., 2002. A framework for multicriteria line generalization to support scientific and engineering modeling. In: Egenhofer, M. J., Mark, D. M. (Eds.), GIScience 2002 Abstracts. pp. 173–175.

van der Poorten, P., Zhou, S., Jones, C. B., 2002. Topologically-consistent map generalisation procedures and multi-scale spatial databases. In: Egenhofer, M. J., Mark, D. M. (Eds.), GIScience 2002. Vol. 2478 of Lecture Notes in Computer Science. Springer, Boulder, CO, USA, pp. 209–227.

Vermeij, M., van Oosterom, P., Quak, W., Tijssen, T., 2003. Storing and using multi-scale topological data efficiently in a client-server DBMS environment. In: Proc. 7th International Conference on GeoComputation.

Weibel, R., 1995. Map generalization in the context of digital systems. Cartography and Geographic Information Systems 22 (4), 259–263.

Weibel, R., 1996. A typology of constraints to line simplification. In: Kraak, M. J., Molenaar, M. (Eds.), Advances in GIS Research II (7th International Symposium on Spatial Data Handling). London, Taylor & Francis, pp. 533–546.