# Constrained set-up of the tGAP structure for progressive vector data transfer

Jan-Henrik Haunert [a,1] — Arta Dilo [b] Peter van Oosterom [b]

[a] *Institut für Kartographie und Geoinformatik, Leibniz Universität Hannover, Appelstraße 9a, 30173 Hannover, Germany*

[b] *OTB, Section GIS-technology, Delft University of Technology, Jaffalaan 9, 2628 BX Delft, The Netherlands*

---

**Abstract**

A promising approach to submit a vector map from a server to a mobile client is to send a coarse representation first, which then is incrementally refined. We consider the problem of defining a sequence of such increments for polygons of different land cover classes in a planar partition. In order to submit well-generalised data sets, we propose a method of two steps: First, we create a generalised representation from a detailed data set, using an optimisation approach that satisfies certain cartographic constraints. Secondly, we define a sequence of basic merge and simplification operations that transforms the most detailed data set gradually into the generalised data set. As each intermediate result defines an intermediate level of detail (LoD), we refer to this procedure as interpolation of LoDs. The obtained sequence of LoDs is stored without geometrical redundancy in the tGAP (topological Generalised Area Partitioning) structure, which is an existing data structure supporting progressive transfer of data. This structure and the algorithm for the interpolation of LoDs have been implemented in an object-relational database and tested for land cover data from the official German topographic data set ATKIS at scale 1:50,000. Results of these tests allow to conclude that the data at lowest LoD and at intermediate LoDs is well generalised. Applying specialised heuristics the applied optimisation method copes with large data sets; the tGAP structure allows to efficiently query and retrieve a data set of an extent and at an LoD defined by the user.

*Key words:* progressive transfer, map generalisation, aggregation

## 1 Introduction

In recent years the Internet has become an important source of digital maps for mobile users. However, applications suffer from bandwidth limitations and restricting devices like small displays. Sending a large-scale map for each request is expensive and time consuming. From a user's perspective this is unsatisfactory if zoom and pan interactions are needed, for example, to first navigate to an area of interest. As this task does not require a map at highest resolution, it is reasonable to send less detailed maps first. In order to define these representations such that characteristic features are preserved, automatic generalisation methods are needed.

In this paper we discuss the generalisation problem in the context of vector data sets for mobile users and focus on the generalisation of polygons in a planar partition representing different land cover types. This data model is commonly used for topographic databases. Generalising such data requires operators for aggregation, collapse and line simplification. In order to explain our motivation and general approach, we first concentrate on the aggregation task; however, we also consider collapse of areas to lines and line simplification in our approach.

Often minimal allowed sizes are defined for polygons at a certain level of detail (LoD), thus generalisation requires to aggregate the polygons in the original data set to satisfy size constraints for a target LoD. An existing approach for this problem is to iteratively select the smallest polygon in the data set and to merge it with its most compatible neighbour until all polygons satisfy the defined thresholds. In fact this procedure does not only yield a data set at a single output LoD, but, in each iteration, also defines an intermediate result. Due to this characteristic, the algorithm has earlier been applied to set up a data structure for progressive data submission: When zooming in, the merge operations simply need to be inverted, in order to gradually refine the data set. We have earlier developed the tGAP (topological

Generalised Area Partitioning) data structure to store the results of this simple generalisation procedure (van Oosterom, 2005); it allows to progressively submit a data set by sending data at a low LoD first and refining the data set by sending increments. We apply the tGAP structure also in this paper; however, we propose a new approach to define the sequence of generalisation steps, in order to improve the quality of generalisation at low LoDs.

Recent research on map generalisation has shown that constraint- and optimisation-based approaches are more flexible and provide better results than classical rule-based approaches, in which conditions are bound to predefined actions (Harrie and Weibel, 2007). In view of this, we developed a method for the generalisation of land cover data based on mixed-integer programming (Haunert and Wolff, 2006), which is a technique for constrained combinatorial optimisation. Figure 1 illustrates the advantages of our method compared to the simple iterative merging procedure. Figure 1(a) shows a sample from the german topographic database ATKIS DLM50, which contains the same amount of details as a topographic map at scale 1:50,000. The sample contains a settlement that is fragmented into several small, non-adjacent polygons. We generalised this data set to satisfy constraints defined for the database ATKIS DLM250, which corresponds to scale 1:250,000. Figure 1(b) shows the result of iteratively merging polygons with their most compatible neighbours, only taking local compatibility measures into account. In this case the settlement is lost, as the small fragments are merged with the adjacent forest polygons. Instead, our optimisation method globally minimises the change of land cover classes. The result is shown in Fig. 1(c): the settlement is kept. We give a more detailed explanation of our optimisation approach in Sect. 3.1.2.

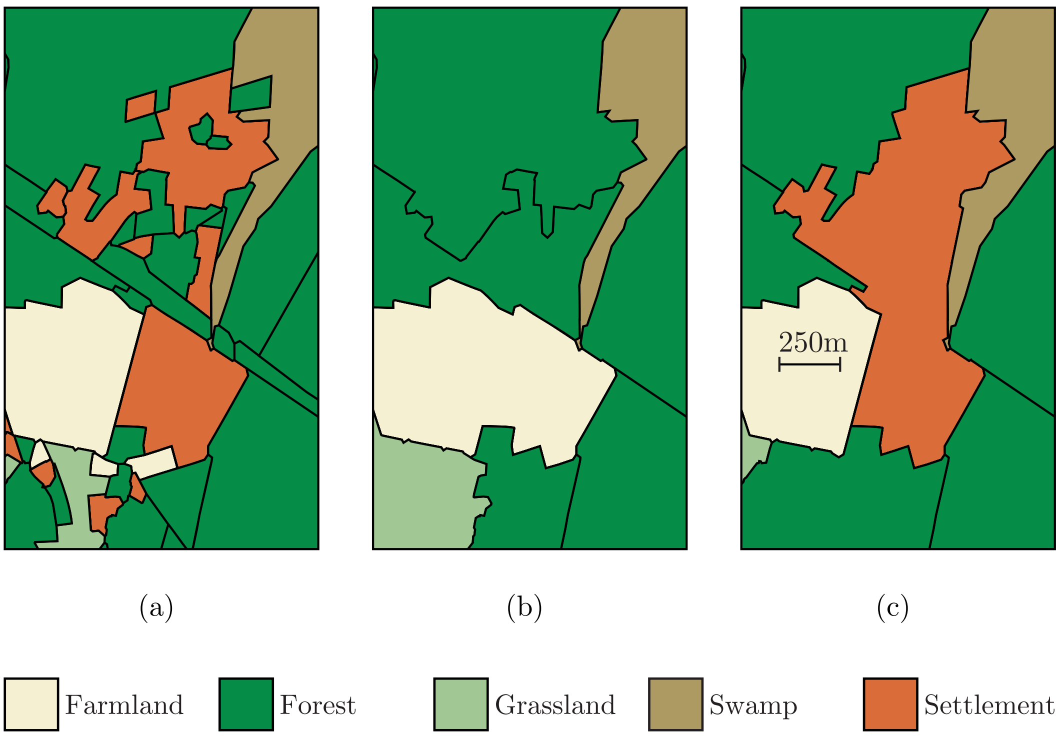


Fig. 1. Comparison of two aggregation methods: (a) Input data set ATKIS DLM50, (b) result of iterative merging, (c) result by optimisation; both results satisfy size constraints defined for the ATKIS DLM250.

In contrast to the iterative merging procedure, optimisation methods for map generalisation normally generate data sets at a single target LoD, that is, they do not define a sequence of data sets that could be used for gradual refinement. Our ambition is to combine the benefits of both approaches: we still wish to provide the data in a hierarchical structure that allows a gradual refinement when zooming in, but would like to be more free in choosing the method to produce representations at low LoDs; in particular, we would like to apply our existing optimisation method. In order to achieve both, we suggest to set up the tGAP structure with two representations at different LoDs: the most detailed data set and a generalised data set, which, for example, can be obtained with our optimisation method. With this input, the iterative algorithm can be controlled to meet the given result or, from a different view, it can be used to interpolate intermediate LoDs. We refer to the obtained structure as *constrained tGAP structure* explained in Section 3. Before introducing this new approach, we review related work of other researchers.

## 2 Related work

### *2.1 Map generalisation*

Map generalisation is the process of deriving a map of smaller scale from a given map. This task is closely related to the definition of cartographic constraints (Beard (1991), Weibel and Dutton (1998)). Violated constraints are commonly referred to as conflicts that need to be resolved by generalisation operators, for example, areas that are too small to be represented in a certain scale need to be aggregated or collapsed (Bader and Weibel, 1997). We distinguish hard and soft constraints: While hard constraints need to be ensured in any case, soft constraints are often contradicting, thus only satisfiable to a certain degree. Consequently, map generalisation is often expressed as optimisation problem, aiming to satisfy soft constraints as much as possible. The optimisation is often done by application of meta-heuristics such as hill-climbing and simulated annealing (Ware and Jones, 1998). In recent years, the application of multi-agent systems has become a mainstream approach (Barrault et al., 2001). This allows to express constraints on map objects and groups of map objects in a generic way. Galanda (2003) discusses this approach in the context of polygon maps, using a hill-climbing strategy for optimisation. Researchers in the field of computational geometry have proposed global and deterministic optimisation approaches, for example, to solve the line simplification problem (de Berg et al., 1998). Often specialised heuristics are needed to find efficient algorithms. We formalised the aggregation task in map generalisation as optimisation problem and also proposed a deterministic approach, which is based on mixed-integer programming (Haunert and Wolff, 2006). The iterative aggregation algorithm that we discussed in Sect. 1 has been applied in different versions by other authors, for example, Jaakkola (1997) use a similar algorithm for the generalisation of raster data with land-cover information. van Smaalen (2003) as well as Cheng and Li (2006) discuss criteria that need to be considered for automated aggregation. According to Timpf (1998) aggregation is the most common type of hierarchy occurring in map series.

### *2.2 Progressive transfer of vector data*

The idea of gradually refining low-resolution vector data in mobile applications has been discussed by several researchers. The refinement of terrain models in computer graphics is presented by De Floriani et al. (2000). Brenner and Sester (2005) present a method to gradually refine building ground plans. As in our first method by iterative aggregation, sequences of refinement increments result from inverted sequences of simplification steps. Bertolotto and Egenhofer (2001) and Follin et al. (2005a) generally express differences between different given vector maps by atomic generalisation and refinement operators. These include the merge operation of two areas, which is sufficient to model the differences in the example of Fig. 1(a) and (c). However, we need to define an appropriate sequence of such pairwise merges, as we intend to also show intermediate results. Methods for the definition of intermediate representations between two scales are proposed by Cecconi (2003) and Merrick et al. (2007). Both are based on morphing algorithms between polygonal lines that allow a smooth animation when zooming in or out. Also the method of Brenner and Sester (2005) includes a morphing technique to give an impression of a continuous process, which is referred to as continuous generalisation. However, these morphing techniques are not used to provide a gradual transformation between two given maps that would allow a progressive refinement. We do not consider the problem of animating discrete steps in a smooth way, thus avoid the term continuous generalisation.

## 3 Map generalisation approach for defining a sequence of LoDs

Our basic assumption is that we are given an algorithm for the classical map generalisation problem, that is, for a given input data set we can produce a data set at any reduced LoD by appropriately setting the parameters of the algorithm. We can apply our optimisation approach for this task or any other generalisation method available. Figure 2 illustrates three different ideas to generate a sequence of LoDs by applying such an algorithm.

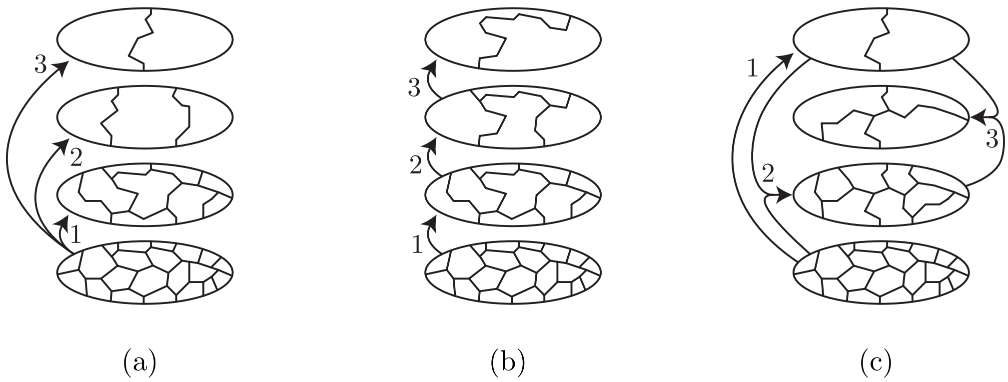


Fig. 2. Approaches to create a sequence of LoDs: (a) generalisation from a single source data set; (b) successive generalisation; (c) interpolation between two data sets.

In Fig. 2(a) the most detailed data set is used as input for the algorithm to generate all levels of the sequence. Though each single data set is well generalised, the obtained sequence of data sets does not conform to the idea of gradual refinement: a single step in the sequence can imply large changes in the data set.

An alternative approach is to generate the sequence of LoDs in small steps, in each step using the previously generated data set as input for the generalisation algorithm (Fig. 2(b)). Though this iterative approach leads to a sequence of relatively small changes between two consecutive LoDs, it entails the risk of generating unsatisfactory results at low LoDs. In particular, this iterative approach does not allow to optimise global quality measures, for example, to minimise changes of land cover classes between the highest LoD and the lowest LoD.

Figure 2(c) shows a third approach, which we propose to overcome the disadvantages of both other methods: We first create the lowest LoD and then define a sequence of intermediate representations (Fig. 2(c)). Using our optimisation method for the first step, we can ensure a well-generalised data set at lowest LoD. In order to define the intermediate LoDs, we can apply a slightly modified version of the iterative algorithm that we have earlier used to set up the tGAP structure. We explain both parts of our method in the remainder of this section.

### 3.1 *Generalisation method for a low LoD*

Our generalisation method for deriving a data set at a low LoD comprises three generalisation operators that we apply in succession: A skeletonisation method that collapses narrow polygons and polygon parts to lines (Haunert and Sester, 2008), an optimisation method that aggregates polygons to satisfy size constraints (Haunert and Wolff, 2006), and an optimisation method for line simplification according to de Berg et al. (1998). Compared to existing generalisation methods, the proposed workflow implies improvements in terms of quality, which are mainly due to the application of optimisation techniques for aggregation. We first present an overview on the applied generalisation workflow and then give a more detailed presentation of the aggregation method.

#### 3.1.1 *Applied generalisation operators*

Figure 3 shows a sample from the input data set after applying the procedures for collapse, aggregation, and line simplification.

Comparing Figures 3 (a) and (b) we observe that the river polygon (blue) in the upper right corner of Fig. 3(a) collapses. This


Fig. 3. Applied generalisation steps: (a) original map , (b) after collapse by skeletons, (c) after aggregation, (d) after line simplification.

is because the polygon's width is lower than 50m – a threshold that we defined according to the resolution of a map at scale 1:250,000. Our procedure, which is based on straight skeletons, also allows to collapse polygon parts, for example, the narrow connection between the main body of the large settlement (red) and the small annex on its left side. Bader and Weibel (1997) presented a similar collapse procedure, which uses a skeleton based on a triangulation of a polygon.

Aggregation is necessary to satisfy size constraints defined for the target LoD, which are given in our case with the specifications of the ATKIS DLM250. In contrast to existing methods our approach is not based on iterative merging of pairs of polygons. Instead, we solve the problem by optimisation, aiming to keep class changes small and to create geometrically compact composite polygons while satisfying hard size constraints. Figure 3 (c) shows the result of this method. Though the settlements in Fig. 3 (b) do not have sufficient size for the target LoD, a settlement of sufficient size is created by including adjacent forest areas; this leads to a solution of little class changes and compact shapes.

Finally, we adapt the line geometries to the target LoD. To solve this task we implemented a line simplification algorithm of de Berg et al. (1998) that defines the simplified line by a subsequence of the original line vertices. The method ensures the bandwidth criterion, that is, for each vertex of the original line the distance to the simplified line must not exceed a user-defined tolerance. Furthermore, the method ensures the simplicity of the planar partition. Subject to these hard constraints the number of vertices in the simplified line is minimised.

#### *3.1.2 Aggregation by optimisation*

We earlier presented our optimisation approach to area aggregation in map generalisation and proved that the problem is NP-hard (Haunert and Wolff, 2006). Due to the absence of efficient exact algorithms, we solved the problem by mixed-integer programming and specialised heuristics. In particular, we introduced a heuristic that allows to decompose arbitrarily large data sets into multiple instances of manageable size (Haunert, 2007a). The results presented in this paper were generated with this approach. However, for the set-up of the tGAP structure with two data sets of different LoD, we do not require the application of a specific optimisation technique, for example, we could also apply meta-heuristics like simulated annealing, which are common in map generalisation (Ware and Jones, 1998). Therefore, we only review the problem definition, including the defined constraints and optimisation criteria. For this we choose a graph-theoretical notation.

We are given a planar graph $G(V, E)$, with $V$ containing a node for each polygon in the original data set and $E$ containing an edge for each two polygons that share a common boundary. We represent the sizes of polygons by weights $w : V \to \mathbb{R}^+$ and their land cover classes by $\gamma : V \to \Gamma$, with $\Gamma$ being the set of all classes. In order to represent minimal allowed sizes for polygons in the target LoD, we introduce the function $\theta : \Gamma \to \mathbb{R}^+$, that is, we allow for different thresholds for different classes. We aim to partition $V$ into mutually disjoint sets $V_1 \cup \ldots \cup V_k = V$. Additionally, we aim to find land cover classes $\gamma'_1, \ldots, \gamma'_k \in \Gamma$. Note that we do not assume that the number $k$ is given in advance. An object in the target scale is defined by a pair $(V_i, \gamma'_i)$. For each such pair we introduce the requirements (hard constraints) that

(1) $V_i$ has weight at least $\theta(\gamma'_i)$,

(2) $V_i$ contains at least one node $v$ with $\gamma(v) = \gamma'_i$,

(3) the subgraph induced by $V_i$ is connected.

We identify two objectives (soft constraints) for the optimisation problem:

(1) Changes of land cover classes should be minimised.

(2) Composite objects should have maximally compact shapes.

In order to express the first objective, we define a cost that is charged when changing a polygon of unit size from one class into another. For this we introduce the cost function $d : \Gamma^2 \to \mathbb{R}_0^+$, whose values could be given explicitly by a quadratic matrix with $|\Gamma| \times |\Gamma|$ elements. The function $d$ can be seen as a semantic distance between classes, that is, it is preferred to change a class to a semantically similar one; different authors have proposed methods to derive such measures from given data models (Rodríguez and Egenhofer, 2004; Yaolin et al., 2002). With these distances, we define the total cost for class change as

$$\sum_{i=1}^{k} \sum_{v \in V_i} w(v) \cdot d(\gamma(v), \gamma_i') \,. \tag{1}$$

To express the compactness of a shape, we tested two different measures (Haunert, 2007b). The first approach is simply to minimise the boundary length of the partition, that is, we charge a cost proportional to the perimeter of a composite region $V_i \subseteq V$:

$$c_p(V_i) = \text{perimeter}(V_i). \tag{2}$$

The second measure is defined as cost for a composite region $V_i$ receiving class $\gamma_i' \in \Gamma$:

$$c_d(V_i, \gamma_i') = \min \left\{ \sum_{v \in V_i} w(v) \cdot \delta(v, u) \;\middle|\; u \in V_i \wedge \gamma(u) = \gamma_i' \right\} \,, \tag{3}$$

with $\delta : V^2 \rightarrow \mathbb{R}_0^+$ being the Euclidean distance between the centroids of two polygons. This means that one node $u \in V_i$ with unchanged class is selected as a reference point and, for each node $v \in V_i$, a cost is charged, which is equal to the product of the size of $v$ and its distance to $u$. As a composite region might contain more than one node with unchanged class, we select the reference point among them, such that the cost is minimal. We refer to such a node as *centre*. Figure 4 illustrates this measure.

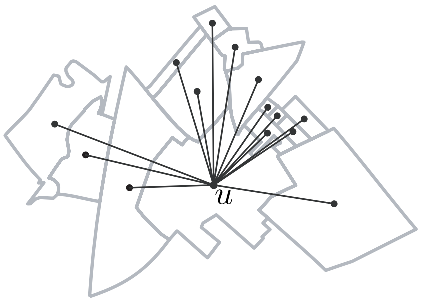


Fig. 4. Compactness according to Equation (3). Node $u$ is selected as centre.

In order to define the trade-off between these criteria, we combine the terms for class change, boundary length and distances to centres in a weighted sum, that is, we minimise

$$\begin{aligned} & s \cdot \sum_{i=1}^{k} \sum_{v \in V_i} w(v) \cdot d(\gamma(v), \gamma_i') \\ & + (s-1) \cdot \sum_{i=1}^{k} \left[ s' \cdot c_p(V_i) + (s'-1) \cdot c_d(V_i, \gamma_i') \right], \end{aligned} \tag{4}$$

with weight factors $s\,, s' \in [0, 1]$.

Applying our method with this cost function and size constraints for the ATKIS DLM250, we automatically generalised a data set of the ATKIS DLM50. This has an extent of 22 km $\times$ 22 km, which corresponds to a complete sheet of a map at scale 1:50,000. The processing took 82 minutes. Compared to the iterative merging procedure we reduced the costs for class change by 20%, the costs for non-compact shapes by 2%, and the overall costs by 8%. We conclude that, applying the developed heuristics, our approach yields high-quality results in modest time. For a more detailed discussion of these tests we refer to our earlier publication (Haunert, 2007a).

### *3.2 Generalisation method for intermediate LoDs*

In order to define data sets at intermediate LoDs, we aim to find a gradual transformation of the data set at highest LoD into the given generalisation result from Sect. 3.1.2. We say that a polygon $a$ is assigned to a polygon $b$, meaning that $a$ is removed from the current data set and the new shape of $b$ becomes the union of both shapes; the class of $b$ is not changed. Formally, this merge operation is represented by the sorted pair $(a, b)$. We seek a sequence of such pairs that yields our generalised data set.

To guarantee that such a sequence exists, we first assure many-to-one relationships between the polygons in both data sets. Since we applied a collapse operator in our generalisation workflow, this condition is not met: some polygons were decomposed into multiple parts, different parts were potentially merged with different neighbours. Therefore, we need to expect many-to-many relationships between polygons at different LoDs. In other words, the generalised data set contains boundaries that were not present in the input data set. In order to ensure many-to-one relationships, we add the additional boundaries in the generalised data set to the data set at highest LoD, that is, we perform a map overlay between the input data set and the data set obtained from the aggregation method. Now there is a sequence of pairwise merges that transforms the input data set into the generalised one. Usually, we have multiple possibilities to define such a sequence.

Our approach to define the sequence of merge operations is similar to the original iterative algorithm. In contrast, when selecting the most compatible neighbour of a polygon, we restrict the set of candidates to polygons in the same composite region. We say that a polygon is active if it still has a neighbour that, in the given result, is contained in the same composite region. For each composite region we define a polygon of unchanged class according to Equation 3 as centre. The following three requirements ensure that a sequence transforms the input map into the given aggregation result:

- The sequence must not be terminated, if there is an active polygon.
- A polygon can only be assigned to neighbours in the same composite region.
- A centre must not be assigned to another polygon.

According to our idea of gradual refinement, we also require that, in each step, the least important active polygon $i$ becomes merged with a neighbour $j$. If $i$ is not a centre, we assign $i$ to $j$, else, to avoid a contradiction with the third requirement, we assign $j$ to $i$. According to the objective in Equation 4 we define $\mathrm{Cost}(a, b)$ to be the cost of the resulting map when assigning $a$ to $b$. The algorithm in Algorithm 1 specifies the approach. In Line 2 the smallest active area of the data set is selected. Lines 5–6 and Lines 8–9 define the merge operations for the cases that $a$ is a centre or not, respectively.

It remains to define intermediate degrees of simplification for lines. We denote a line at highest LoD by $l_1$, its vertices by $V_1$, and

**Algorithm 1** Generation of intermediate LoDs

**while** there is an active polygon **do**
  $a \leftarrow$ smallest active polygon
  $N \leftarrow$ neighbours of $a$ in the same composite region
  **if** $a$ is a centre **then**
    $b \leftarrow b' \in N$ with minimum $\texttt{Cost}(b', a)$
    assign $b$ to $a$.
  **else**
    $b \leftarrow b' \in N$ with minimum $\texttt{Cost}(a, b')$
    assign $a$ to $b$
  **end if**
**end while**

the simplified line at lowest LoD by $l_2$, having vertices $V_2 \subseteq V_1$. To define intermediate LoDs we split $l_1$ into multiple lines, each corresponding to a single line segment of $l_2$. Simplifying these lines with parameters for the intermediate scale, we obtain a set of vertices $V$ such that $V_1 \supseteq V \supseteq V_2$, thus an intermediate representation that allows a refinement by adding vertices when zooming from low to high LoD. We have applied this procedure on the client side to produce the presented results, but also could store the resulting hierarchy of vertices on the server side.

Using Algorithm 1 and the explained procedure for intermediate line simplification levels, we obtain intermediate representations as shown in Fig. 5. The sequence only implies small changes in each step and terminates with the well-generalised data set obtained from our optimisation method. In the next section we explain how to represent this sequence in the tGAP structure.

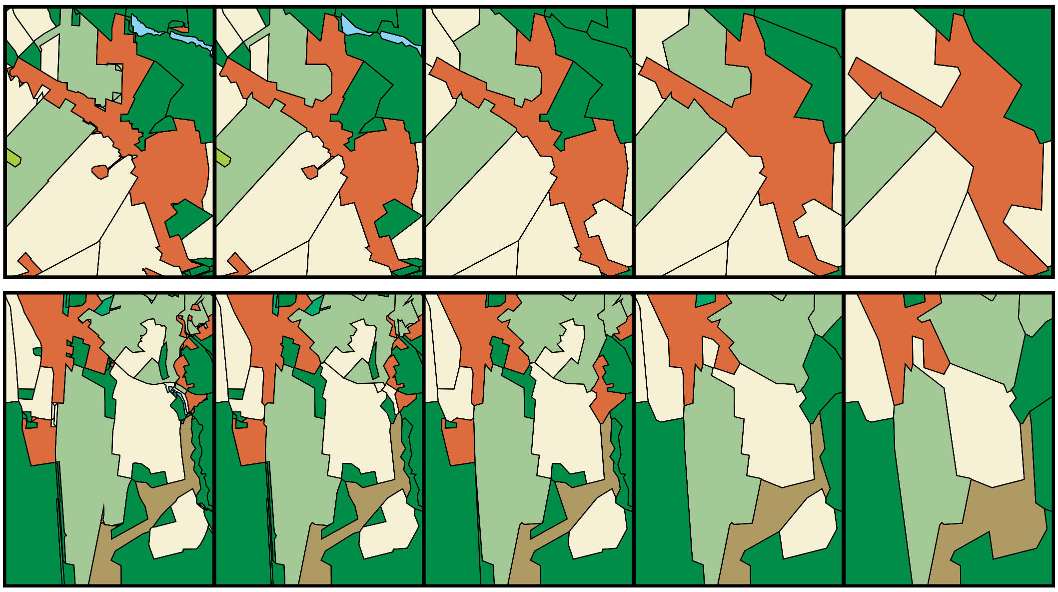

Fig. 5. Two examples processed with Algorithm 1. From left to right: Original map, found intermediate representations, and map generalised by optimisation.

## 4 The tGAP structure

The tGAP structure is a collection of data structures that store the results of a generalisation performed in a preprocessing step. The data structures support a vario-scale representation of a planar partition without redundancy of geometry. Area features at the highest level of detail (LoD)⋆ are stored using a topological model. There is no redundancy of geometry in this level as the shared boundary edges between neighbour faces are stored only

⋆ We use the terms LoD and map scale interchangeably.

once. The generalisation process reduces the level of detail by merging features (see Figure 6). For features created from generalisation references are stored to the composing features of higher detail level. The data structures provide the features to be shown at any required LoD, thus enabling an on-the-fly generalisation by feature selection.

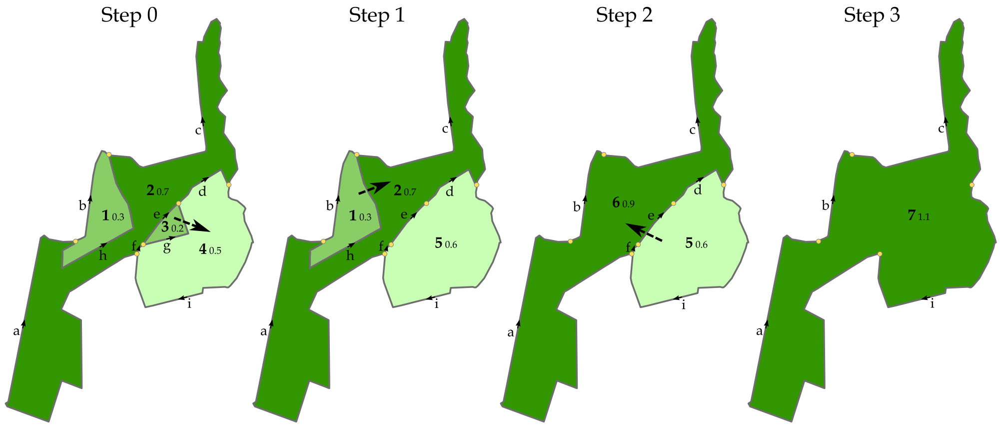


Fig. 6. Steps of the generalisation process for the partition shown at ‘Step 0’. Faces are numbered, and edges are labelled with letters. The subscript to a face number is its importance value.

The off-line generalisation that fills the tGAP is an iterative process. Figure 6 illustrates the generalisation process for the planar partition shown in ‘Step 0’; the other maps show the result of each generalisation step. In each step, the least important area feature is merged to its most compatible neighbour. A dashed arrow shows the least important face for the current step, and its most compatible neighbour (where the arrow is headed). In the next step, the least important face is merged to its most compatible neighbour. The process continues until all is merged to one face.

#### *4.0.1 Building tGAP structure*

The result of each generalisation step is again a planar partition, which is a collection of faces. A face is constructed by the set of edges that form its boundary. The collection of faces at a certain generalisation step determines the collection of edges that form the partition at this step. There is a last issue in the generalisation process: boundary edges get simplified as the level of detail decreases. To capture the generalisation process we need to keep track of face merging, how this is reflected to boundary edges, and the simplification of edges. The data structures forming the tGAP take care of these three issues. The tGAP structure consists of a structure holding the hierarchy of faces, an edge forest that holds the corresponding relations between boundary edges, and BLG (Binary Line Generalisation) trees, for each edge one tree, which stores the result of the Douglas-Peucker algorithm (Douglas and Peucker, 1973) for line simplification. There is a trade-of decision between storing BLGs and do simplification from BLG reading, or storing edges geometry and perform line simplification online. Here we chose for online line simplification. For the complete treatment of BLGs we refer the interested readers to (van Oosterom and Schenkelaars, 1995; van Oosterom, 2005; Meijers, 2006).

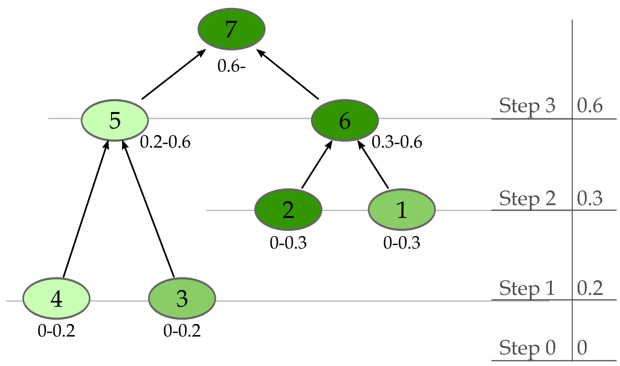


Fig. 7. The face tree corresponding to the generalisation of Figure 6. Nodes in the tree are faces, and lines depict merging of two faces into the parent face.

Generalisation starts with the original (i.e., highest LoD) faces. A generalisation step merges two neighbour faces to a new one, which continues further in the generalisation process. The new face and the merged faces have a parent-child relation. The hierarchy of faces created by this process is a tree. Leaf nodes are the original faces, the root is the full map extent. Figure 7 shows the face hierarchy created by the generalisation process of Figure 6. In the right side of the tree are shown the steps performed to create the tree, each step is associated with its importance value.

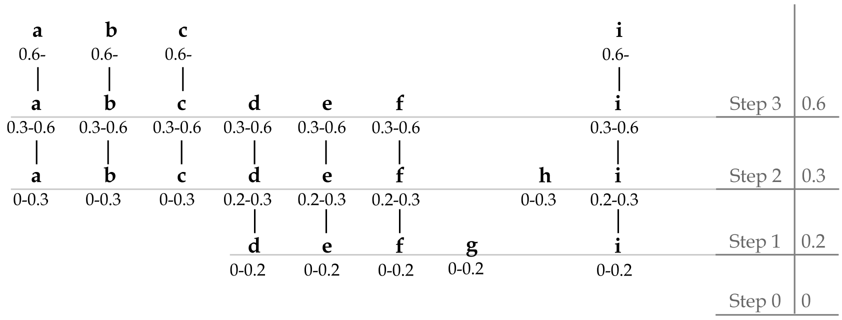


Fig. 8. The edge forest for the generalisation process of Figure 6.

We store faces using the left-right topology without edge references. This model stores the edge geometry (as a directed polyline with start and end node references), together with references to the left and right face of the edge. Each face is constructed from the list of edges that refer to it as a left or right face. That determines the type of changes an edge undergoes in the generalisation process. An edge disappears if it is part of the common boundary of the two merged faces, e.g., edge ‘g’ in Step 1 (see Figure 6). The other edges may continue existing, but the left or right face of each edge is changed, e.g. edges ‘d’, ‘e’, ‘f’, and ‘i’ in Step 1. An edge takes the importance value from the importance of the step in which it changed. Figure 8 shows the complete edge hierarchy for the generalisation process of Figure 6. On the right side there are the steps at which changes occur, each step associated with its importance level.

#### 4.0.2 *Using tGAP structure*

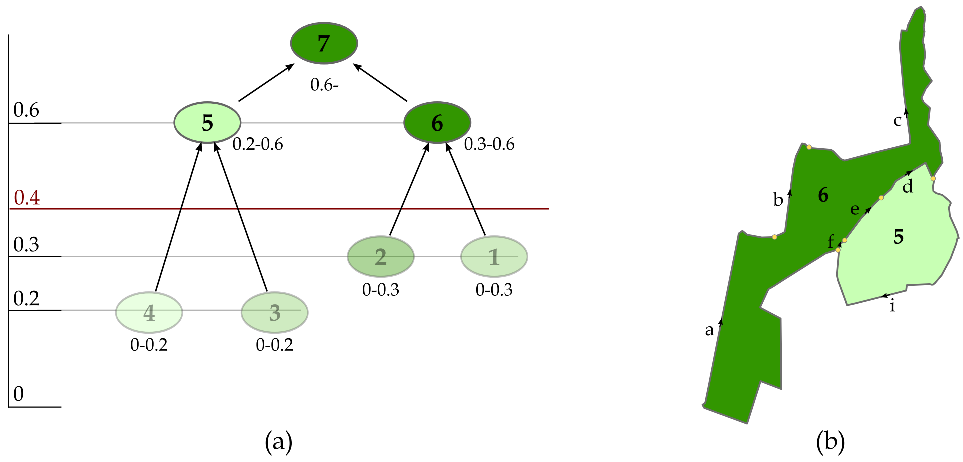


Fig. 9. Selection in the face tree: faces to appear at the importance value = 0.4.

After the tGAP structure is built, it can be used to select features to be shown for a certain scale. A given map scale is translated to importance value, which is used to select features. A face will be shown if the given importance value is in the importance range of the face. Figure 9 gives faces to be shown for an importance value equal to 0.4. This importance value is in the range [0.3, 0.6) formed between steps 2 and 3. The map created from Step 2 is unchanged for values in this range. Faces to be shown are the leaf nodes of the (sub)tree created by cutting all nodes with importance values lower than 0.4; these are faces 5 and 6.

Edges to be shown at the importance value 0.4 are those that include this value in their importance range. They are the boundaries of faces to be shown for that importance. Figure 10 shows

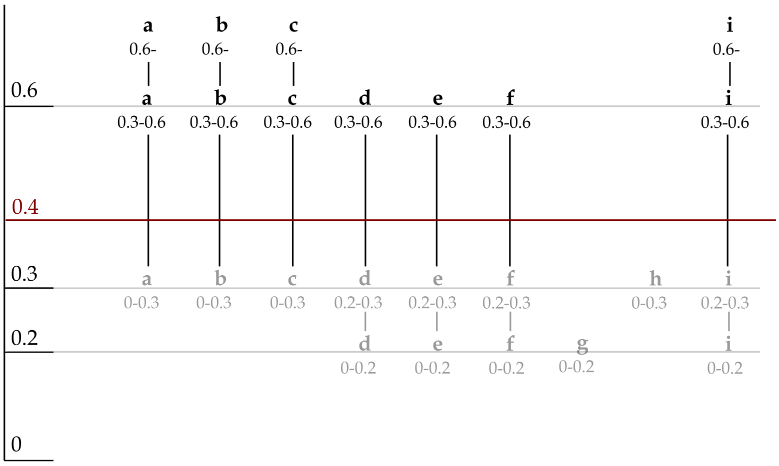


Fig. 10. Selection of edges to appear at the importance value = 0.4.

the edges to be displayed at the importance value 0.4. The edges are the leaf nodes in the forest remained after cutting nodes with importance less than 0.4.

### 4.1 Implementation in Oracle Spatial

The constrained tGAP information is stored in Oracle Spatial. The Oracle tables and their relationships are shown in Figure 11. Arrows associating tables show foreign key relationships. Table `Face` holds information about faces: the centroid needed from the optimisation algorithm, the class to which it belongs, the small scale aggregate region to which it is constrained, centre which takes only values 1 and 0, 1 meaning the face is a centre, the area size used to calculate the cost of merging, the importance range, and its parent (in the face tree). Information about edges is split in two tables: **EdgeGeo** that stores the geometry, its length needed for cost calculation, and references to start and end node; **EdgeLOD** that stores references to left, and right face as they change during the generalisation (while the geometry remains the same), and the corresponding importance ranges. **Node** table stores the geometry of nodes. Table **ClassInfo** stores information about classes: code as referred in **Face** table, name and description, as well as weight needed for calculation of importance values of faces. Table **ClassCompatibility** stores the compatibility values as **cost** of changing from the **from_class** to the **to_class**.

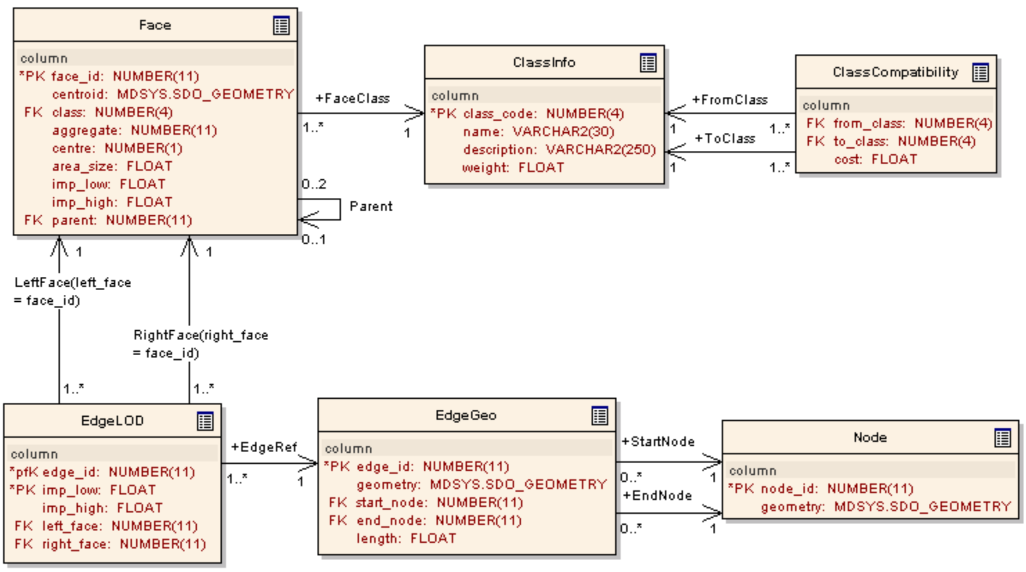


Fig. 11. UML diagram of tables and relationships that store the tGAP information in Oracle Spatial.

## 5 Progressive transfer

In our approach edges are sent progressively from the server to the client. Edge information sent by the server consists of edge geometry, scale and topological information, i.e., importance range as well as references to left-right faces and start-end nodes. The server also sends thematic information about faces, but not their geometry. The client builds topology, i.e., creates geometry of faces, and visualises faces with their thematic information.

There are different situations in the server-client communication: the initial state, panning, zoom-in, and zoom-out operations. In all the situations the client provides a spatial extent and a scale to the server. The initial state is the first request from the client side. For an initial state requesting data, e.g., at the importance value `0.7`, the SQL queries on the server are:

```
select g.edge_id, i.left_face, i.right_face,
    g.start_node, g.end_node, g.geometry
from EdgeLOD i, EdgeGeo g
where i.edge_id = g.edge_id and
    i.imp_low <= 0.7 and i.imp_high > 0.7
order by i.imp_high desc;

select face_id, class, region
from Face
```

```
where imp_low <= 0.7 and imp_high > 0.7
order by imp_high desc;
```

The first query collects edge information, the second collects face information. Additionally, both queries should consider the spatial extent (in the where clause). Ordering by `imp_high` allows sending edges according to their importance, i.e., edges that are visible at smaller scales will be sent first. Queries for pan and zoom-out operations are similar to the above, the only difference being in the spatial extent request.

A zoom-in operation on the client side requires refinement of the already received information. Features that appear at larger scales have lower importance values (see Figures 9 and 10). This requires faces and edges whose importance is lower than the current importance. To get additional information, e.g., for the range $[0.2, 0.7[$, we collect separately the geometry of edges that appear for the first time in this range, topological information for edges to appear at importance value $0.2$ as well as face information for this last value. The first query below collects geometry of new edges, the second collects topology information of edges to appear at importance $0.2$, and the third collects thematic information of faces.

```
select edge_id, start_node, end_node, geometry
from EdgeGeo
where edge_id in
    (select edge_id
    from EdgeLOD
    group by edge_id
    having min(imp_high) > 0.2 and max(imp_high) <= 0.7);
```

```
select edge_id, left_face, right_face
from EdgeLOD
where imp_low <= 0.2 and imp_high > 0.2;
```

```
select face_id, class, region
from Face
where imp_low <= 0.2 and imp_high > 0.2;
```

The client visualises edges as they come, but has to wait until all the information for the requested range is sent, then builds the topology of faces and visualises them.

We may send information in small packages, one package containing the information about two faces merged in a step. We collect the information for each step with importance in the requested range. The importance values of all these steps are collected by :

```
select distinct(imp_high) as step_imp
from Face
where imp_high > 0.2 and imp_high <= 0.7
order by imp_high desc;
```

Information for one package is collected from similar queries with the above, replacing the ‘having’ condition in the first query with `max(imp_high) = step_imp`, and the ‘where’ condition in the last two queries with `imp_high = step_imp`. A drawback of this solution is that a lot of queries and requests have to be send from the client to the sever. First of all: this causes overhead, but perhaps more serious, due to network delays it is not sure that all answers are also received in the proper order. A specific communication channel supporting ’streaming’ data has to be used. On the server side still the original queries are executed (including the order by), but before sending the query results to the client, smaller streaming packages are created. One package contains the used edges and the faces involved in a step of the tGAP structure creation: two faces are merged and at least one edge is removed. The streaming communication at the client side will also make sure that the packages are processed in the right order. In case a package is missing due to a delay, the client waits for it before processing others.

## 6 Future work

Our method is based on the assumption that the aggregation is the dominating relationship between features of two given data sets. Additional lines resulting from the collapse operation explained in (Haunert and Sester, 2008) are simply included in the original (large scale) map. Using the proposed skeleton operator, the overhead is limited, but, if we applied more generalisation operators like displacement, exaggeration, and typification, this will result in more additional edges and faces. Instead of having the collapse operation (and other operations) only available as preprocessing operation, it might also be fully integrated in the tGAP structure. The effect of including the area-to-line collapse function is that the tGAP face-tree will become a tGAP face-DAG (directed acyclic graph) as the collapsed face will be partitioned and assigned to multiple parents. However, this will happen at most only once for every face (area object) and does fit well in the proposed table structure after a slight modification in change of cardinality of the parent attribute in the face table.

Until now we do not have empirical results concerning the inclusion of additional operators. For the future, we plan to perform tests on a data set of realistic size. We also plan to test our method with two different settings for the line simplification. This can either be done by progressively sending the (stored) BLGs to the client or by sending the full edge geometry to the client and performing the line simplification on-the-fly.

As discussed at the end of Section 5, additional research is needed concerning the use of streaming protocols and the appropriate size of submitted packages.

## 7 Conclusion

We have presented a new approach to set up a data structure for the progressive submission of vector maps. Our idea is to first generalise the large scale map to a much smaller scale (of optimised high quality) and, in a second step, to find a sequence of basic merge operations that enables a gradual transformation between both representations. Though we used a simple iterative algorithm for the second task, our approach ensures a well-generalised map at small scale; this is often needed for navigation tasks, e.g., to pan to the user's area of interest.

We have shown how to cope with aggregation and line simplification and suggested a simple way to also consider area to line (or point) collapse. Generally, we do not restrict to any certain generalisation method in the first preprocessing step. The main contribution of the paper is that it demonstrates an approach to have a good quality variable scale structure. The unconstrained tGAP structure may result in less quality medium and small scale representations. Using constraints, either computed (see Section 3.1.2) or from other medium/small scale source, will guarantee that the quality at the constraint scale is obtained (and that the quality at the intermediate scales is improved based on the conducted visual inspection). Besides the improved quality there are two important additional characteristics for the constrained tGAP structure: 1. it does not contain any geometric redundancy (and only minimal multiple representations of a feature; e.g. at most once for an area to line collapse) and 2. it does support progressive transfer of vector data to be used in smooth zoom functionality at the client side.

## References

Ai, T., Li, Z., Liu, Y., 2005. Developments in Spatial Data Handling. Springer, Berlin, Germany, Ch. Progressive Transmission of Vector Data Based on Changes Accumulation Model, pp. 85–96, in Proceedings of 11th International Symposium on Spatial Data Handling.

Bader, M., Weibel, R., 1997. Detecting and resolving size and proximity conflicts in the generalization of polygonal maps. In: Proceedings of the 18th International Cartographic Conference (ICC'97), Stockholm, Sweden. pp. 1525–1532.

Barrault, M., Regnauld, N., Duchene, C., Haire, K., Baeijs, C., Demazeau, Y., Hardy, P., Mackaness, W., Ruas, A., Weibel, R., 2001. Integrating multi-agent, object-oriented, and algorithmic techniques for improved automated map generalization. In: Proc. of the 20th International Cartographic Conference, Beijing, China. Vol. 3. pp. 2110–2116.

Beard, M. K., 1991. Constraints on rule formulation. In: Buttenfield, B. P., McMaster, R. B. (Eds.), Map Generalization: Making rules for knowledge representation. Longman Scientific & Technical, Ch. 7, pp. 121–149.

Bertolotto, M., August 2007. Spatial Data on the Web. Springer Berlin Heidelberg, Ch. Progressive Techniques for Efficient Vector Map Data Transmission: An Overview, pp. 65–84.

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

Bo, A., Ai, T., Xinming, T., 2008. Progressive transmission of vector map on the map. In: Proceedings of the XXIst ISPRS Congress, July 3–11, 2008, Beijing, China. No. XXXVII(Part B2) in International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences. pp. 411–418.

Brenner, C., Sester, M., July 2005. Continuous generalisation for small mobile displays. In: Next Generation Geospatial Information. Vol. 1. Routledge, UK, pp. 33–42.

Buttenfield, B. P., 2002. Transmitting vector geospatial data across the internet. In: Proceedings of the Second International Conference on Geographic Information Science (GIScience'02), September 25 - 28, 2002. Vol. 2478 of Lecture Notes In Computer Science. Springer, Berlin, Germany, pp. 51–64.

Cecconi, A., 2003. Integration of cartographic generalization and multi-scale databases for enhanced web mapping. Ph.D. thesis, University of Zurich, Switzerland.

Cheng, T., Li, Z., 2006. Toward quantitative measures for the semantic quality of polygon generalization. Cartographica 41, 135–147.

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

De Floriani, L., Magillo, P., Morando, F., Puppo, E., 2000. Dynamic view-dependent multiresolution on a client–server architecture. Computer-Aided Design 32 (13), 805–823.

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.

Follin, J.-M., Bouju, A., Bertrand, F., Boursier, P., 2005a. Multiresolution extension for transmission of geodata in a mobile context. Compunters & Geosciences 31, 179–188.

Follin, J.-M., Bouju, A., Bertrand, F., Stockus, A., 2005b. Web and Wireless Geographical Information Systems. Vol. 3833 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg, Ch. An Increment Based Model for Multi-resolution Geodata Management in a Mobile System, pp. 42–53, in Proceedings of Web and Wireless Geographical Information Systems, W2GIS'05.

Galanda, M., 2003. Automated polygon generalization in a multi agent system. Ph.D. thesis, Department of Geography, University of Zurich, Switzerland.

Harrie, L., Weibel, R., 2007. Modelling the overall process of map generalization. In: Mackaness, W., Ruas, A., Sarjakoski, T. L. (Eds.), Generalisation of geographic information: Cartographic modelling and applications. Elsevier, Oxford, UK, Ch. 4, pp. 67–88.

Haunert, J.-H., 2007a. Efficient area aggregation by combination of different techniques. In: Proc. of 10th ICA Workshop on Generalisation and Multiple Representation, 2–3 August 2007, Moscow, Russia.

Haunert, J.-H., 2007b. A formal model and mixed-integer program for area aggregation in map generalization. In: Proc. Photogrammetric Image Analysis (PIA07), IntArchPhRS XXXVI, PART 3/W49A. ISPRS, pp. 161–166.

Haunert, J.-H., Sester, M., 2008. Area collapse and road centerlines based on straight skeletons. GeoInformatica 12 (2), 169–191.

Haunert, J.-H., Wolff, A., 2006. Generalization of land cover maps by mixed integer programming. In: Proc. 14th Int. ACM Sympos. Advances in Geographic Information Systems (ACM-GIS

’06). pp. 75–82.

Jaakkola, O., 1997. Quality and automatic generalization of land cover data. Ph.D. thesis, Department of Geography, University of Helsinki.

Meijers, M., June 2006. Implementation and testing of variable scale topological data structures. Master’s thesis, TU Delft.

Merrick, D., Nöllenburg, M., Wolff, A., Benkert, M., 2007. Morphing polygonal lines: A step towards continuous generalization. In: Winstanley, A. (Ed.), Proc. 15th Annu. Geograph. Inform. Sci. Research Conf. UK (GISRUK’07). Maynooth, Ireland, pp. 390–399.

Monmonier, M., 1989. Regionalizing and matching features for interpolated displacement in the automated generalization of digital cartographic databases. Cartographica 26 (2), 21–39.

Rodríguez, M. A., Egenhofer, M. J., 2004. Comparing geospatial entity classes: an asymmetric and context dependent similarity measure. International Journal of Geographical Information Science 18 (3), 229–256.

Sester, M., Brenner, C., 2005. Developments in Spatial Data Handling. Springer, Ch. Continuous Generalization for Visualization on Small Mobile Devices, pp. 355–368, in Proceedings of

11th International Symposium on Spatial Data Handling.

Timpf, S., 1998. Hierarchical structures in map series. Ph.D. thesis, Technical University Vienna, Austria.

van Oosterom, P., 2005. Variable-scale topological data structures suitable for progressive data transfer: the gap-face tree and gap-edge forest. Cartography and geographic information science 32, 331–346.

van Oosterom, P., Schenkelaars, V., 1995. The development of an interactive multi-scale gis. International Journal of Geographical Information Systems 9 (5), 489–507.

van Smaalen, J. W. N., 2003. Automated aggregation of geographic objects. Ph.D. thesis, Wageningen University, The Netherlands.

Ware, J. M., Jones, C. B., 1998. Conflict reduction in map generalization using iterative improvement. GeoInformatica 2 (4), 383–407.

Weibel, R., Dutton, G., 1998. Constraint-based automated map generalization. In: 8th International Symposium on Spatial Data Handling. pp. 214–224.

Yang, B., June 2005. A multi-resolution model of vector map data for rapid transmission over the internet. Computers &

Geosciences 31 (5), 569–578.

Yang, B., Purves, R., Weibel, R., 2005. Efficient transmission of vector data over the internet. International Journal of Geographical Information Science 21 (2), 215–237.

Yaolin, L., Molenaar, M., Kraak, M.-J., 2002. Semantic similarity evaluation model in categorical database generalization. In: Proceedings of the ISPRS Commission IV Symposium "Geospatial Theory, Processing and Applications", July 9–12, 2002, Ottawa, Canada. No. XXXIV(Part 4) in International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences.

Zhou, M., Bertolotto, M., 2004. A data structure for efficient transmission of generalised vector maps. In: Bubak, M., van Albada, G. D., Sloot, P. M., Dongarra, J. (Eds.), Computational Science – ICCS 2004. Vol. 3039 of Lecture Notes in Computer Science. Springer-Verlag Berlin Heidelberg, pp. 948–955.