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

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

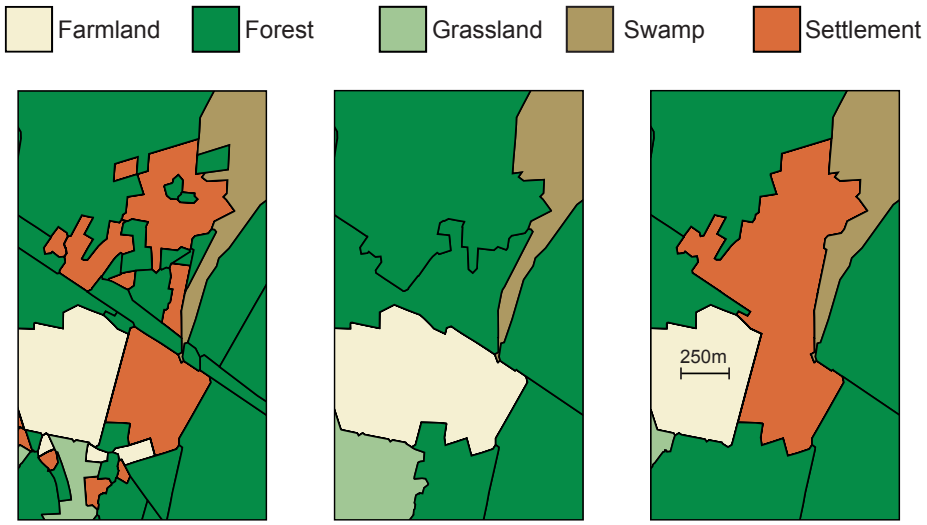
[a] *Institut für Kartographie und Geoinformatik, Leibniz Universität Hannover, Appelstraße 9a, 30167 Hannover, Germany*
[b] *OTB, Section GIS-technology, Delft University of Technology, Jaffalaan 9, 2628 BX Delft, The Netherlands*

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 areas of different land cover classes in a planar partition. In order to submit well-generalised datasets, we propose a method of two stages: First, we create a generalised representation from a detailed dataset, 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 dataset gradually into the generalised dataset. The obtained sequence of gradual transformations is stored without geometrical redundancy in a structure that builds up on the previously developed tGAP (topological Generalised Area Partitioning) structure. This structure and the algorithm for intermediate levels of detail (LoD) have been implemented in an object-relational database and tested for land cover data from the official German topographic dataset ATKIS at scale 1:50 000 to the target scale 1:250 000. Results of these tests allow us 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 datasets; the tGAP structure allows users to efficiently query and retrieve a dataset at a specified LoD. Data are sent progressively from the server to the client: First a coarse representation is sent, which is refined until the requested LoD is reached.

Key words: map generalisation, aggregation, vector data refinement, progressive transmission

*Corresponding author. Tel.: +49 511 76219369; fax: +49 511 7622780.
Email addresses: jan.haunert@ikg.uni-hannover.de (Jan-Henrik Haunert), A.Dilo@tudelft.nl (Arta Dilo), P.J.M.vanOosterom@tudelft.nl (Peter van Oosterom)



(a) input dataset (b) iterative merging (c) optimisation

Figure 1: Comparison of two aggregation methods.

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 datasets for mobile users and focus on the generalisation of areas in a planar partition. This kind of representation is often used for topographic data. Generalising such data requires different operators, for example, aggregation, collapse, and line simplification.

Figure 1 illustrates a common generalisation problem. The sample in Fig. 1(a) was taken from the German topographic database ATKIS DLM 50, which contains the same amount of details as a topographic map at scale 1:50 000. In order to generalise these data, we need to satisfy size constraints defined for the lower level of detail (LoD) that we aim to achieve. In many countries, such constraints are defined in specifications of topographic databases, for example,

in Germany[1], Canada[2], and Australia[3].

Obviously, size constraints can be satisfied by aggregation, that is, by combining the input areas into fewer composite regions. This can be done, for example, by iteratively selecting the least important area in the dataset and merging it with its most compatible neighbour until all areas have sufficient size. Different measures of compatibility and importance have been proposed (Cheng and Li, 2006; van Oosterom, 2005). Figure 1(b) shows the result of the algorithm when applying size constraints defined for the ATKIS DLM 250, which corresponds to scale 1:250 000. Though all areas in the output have sufficient size, the example reveals a shortcoming of the iterative algorithm: As the algorithm only takes the compatibility between adjacent areas into account, large parts of the dataset change their class. In particular, the group of non-adjacent settlement areas is lost. This is because each single settlement area is too small for the target scale and becomes merged with a neighbour of another class, for example, forest. However, together the group of settlements would reach the required size. In order to preserve such groups, we developed an optimisation method for aggregation that minimises the change of land cover classes while ensuring contiguous regions of sufficient size (Haunert and Wolff, 2006). The result is shown in Fig. 1(c): The small settlements are grouped into one large composite region. Generally, constraint- and optimisation-based approaches to map generalisation are considered highly flexible and capable of providing high-quality results (Harrie and Weibel, 2007).

Though the simple iterative algorithm cannot offer results of the same quality as the optimisation method, it has a feature that is useful for progressive data submission: The algorithm does not only yield a dataset at a single output LoD, but, in each iteration, also defines an intermediate result. When zooming in, the merge operations simply need to be inverted, in order to gradually refine the dataset. We have earlier developed the tGAP (topological Generalised Area Partitioning) structure to store the results of the gradual generalisation process (van Oosterom, 2005); with this it is possible to progressively submit a dataset by sending data at a low LoD first and refining the dataset by sending increments.

In contrast to the iterative merging procedure, optimisation methods for map generalisation normally generate datasets at a single target LoD, that is, they do not define a sequence of datasets 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 for a gradual refinement when zooming in, but would like to be more free in choosing the method to produce representations at low LoDs. For example, we would like to apply

[1] Arbeitsgemeinschaft der Vermessungsverwaltungen der Länder der Bundesrepublik Deutschland: ATKIS-Objektartenkatalog. `http://www.atkis.de` [Accessed 09 January 2009]

[2] National Resources Canada: National Topographic Data Base, Data Dictionary. `http://ftp2.cits.rncan.gc.ca/pub/bndt/doc/dictntd3_en.pdf` [Accessed 09 January 2009]

[3] Geoscience Australia: Topographic Data and Map Specifications. `http://www.ga.gov.au/mapspecs/250k100k/` [Accessed 09 January 2009]

our existing optimisation method or any other method available. In order to achieve this, we suggest to set up the tGAP structure with two representations at different LoDs: the most detailed dataset and a generalised dataset, which, in this paper, is 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 define intermediate LoDs.

The rest of this paper is structured as follows. In Sect. 2 we review related work. Section 3 presents our generalisation approach to define a sequence of LoDs that allows for a gradual refinement of the lowest LoD into the highest LoD. In Sect. 4 we introduce the *constrained tGAP structure* that allows the generalisation process to be recorded. We show how to use this structure for reconstructing a required LoD and for progressive submission of data. Section 5 gives an outlook on future research and concludes the paper.

2. Related work

In this section we discuss related work on map generalisation in general (Sect. 2.1) and, in particular, in the context of progressive transfer of vector data (Sect. 2.2).

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). In the map generalisation literature, it is often mentioned that constraints are contradicting, thus can only be satisfied to a certain degree. For example, Galanda (2003) notes that "map generalization has to compromise between different competing needs (i.e., generalization constraints)". In mathematics, the term constraint has another meaning, that is, a constraint can be satisfied or not, but there is no state in between (Papadimitriou and Steiglitz, 1998). In the following we use the term constraint with this meaning and refer to potentially competing needs as *objectives*, for example, changes of land cover classes should be small.

The optimisation of objectives in map generalisation 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 different generalisation operators to be integrated. 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

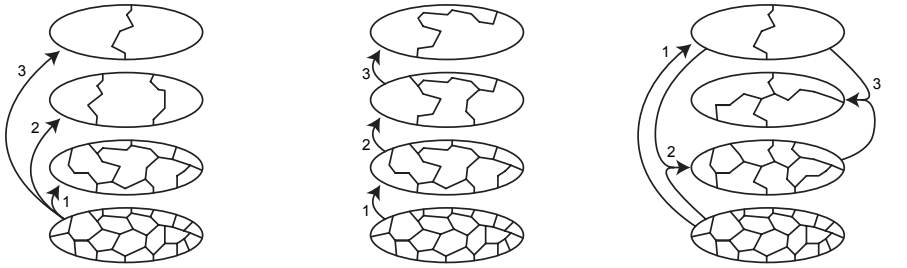
as optimisation problem and also proposed a deterministic approach, which is based on mixed-integer programming (Haunert and Wolff, 2006). The iterative aggregation algorithm that we discussed in Sect. 1 has been applied in different versions by other authors, for example, Jaakkola (1997) uses a similar algorithm for the generalisation of raster data with land-cover information. Cheng and Li (2006) as well as van Smaalen (2003) discuss criteria that need to be considered for automated aggregation.

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). Buttenfield (2002) presents an algorithm for the gradual refinement of polyline coordinates; the method is based on the line simplification algorithm of Douglas and Peucker (1973) and assures that topological properties are preserved. Brenner and Sester (2005) present a method to gradually refine building ground plans. As in our first method by iterative aggregation, a sequence of refinement increments results from an inverted sequence of simplification steps. Operators for simplification and reconstruction of simplified data are proposed by Yang (2005) and Yang et al. (2005). Simplification is performed with a constrained removal of vertices; a consistent topology is kept by enforcing a set of rules. Reconstruction operators restore the original data from the simplified versions.

Ai et al. (2005) propose a changes accumulation model, which sees the difference between two consecutive representations as an addition or subtraction of change patches, therefore the difference between a source and a target scale as an accumulation of such changes. Taking the example of a river network, Bo et al. (2008) present a structure that is based on a changes accumulation model, which is used for progressive transfer. Bertolotto and Egenhofer (2001) and Follin et al. (2005) 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 for a smooth animation when zooming in or out. The interpolated generalisation proposed by Monmonier (1989) follows a similar approach: it uses a smaller scale representation of data to guide the generalisation at intermediate scales. First, matching between features from the two scales should be performed; this is followed by the interpolated displacement and simplification. 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 for a progressive refinement. We do not con-



(a) generalisation from a single source dataset

(b) successive generalisation

(c) defining intermediate LoDs

Figure 2: Approaches to create a sequence of LoDs.

sider 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 dataset we can produce a dataset 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.

In Fig. 2(a) the most detailed dataset is used as input for the algorithm to generate all levels of the sequence. Though each single dataset is well generalised, the obtained sequence of datasets does not conform to the idea of gradual refinement, for example, a line boundary that appears at a smaller LoD may not be present at a higher LoD.

An alternative approach is to generate the sequence of LoDs in small steps, in each step using the previously generated dataset as input for the generalisation algorithm (Fig. 2(b)). The iterative method that we previously used to set up the tGAP structure follows this approach. Though it leads to a sequence of relatively small changes between two consecutive LoDs, it entails the risk of getting unsatisfactory results at low LoDs. In particular, this iterative approach does not allow us 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 dataset at lowest LoD and then define a sequence of intermediate representations (Fig. 2(c)). Using our optimisation method for the first stage, we can ensure a well-generalised dataset at lowest LoD. In order to define the intermediate LoDs, we apply a modified version of the iterative algorithm that we earlier applied to set up the tGAP structure. We present both stages of our method in the remainder of this section.

3.1. Generalisation method for the lowest LoD

Our generalisation method for deriving the dataset at the lowest LoD comprises three generalisation operators that we apply in succession: A skeletonisation method that collapses narrow areas and area parts to lines (Haunert and Sester, 2008), an optimisation method that aggregates areas to satisfy size constraints (Haunert and Wolff, 2006), and an optimisation method for line simplification (de Berg et al., 1998). Figure 3 shows a sample from the input dataset ATKIS DLM 50 that we generalised with this work flow according to specifications for the ATKIS DLM 250. We refer to the input dataset as S_{highLoD}. Applying the collapse operator to S_{highLoD} we obtain S_{coll}, applying the aggregation operator to S_{coll} we obtain S_{aggr}, and finally, applying the line simplification operator to S_{aggr} we obtain the generalised dataset S_{lowLoD}.

Comparing Figures 3(a) and 3(b) we observe that the river area (blue) in the upper right corner of Fig. 3(a) collapses. This is because the area's width is lower than 50 m – 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 us to collapse area 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 in our case are given with the specifications of the ATKIS DLM 250. In contrast to existing methods, our approach is not based on iterative merging of pairs of areas. Instead, we solve the problem by optimisation, aiming to keep class changes small and to create geometrically compact composite regions while satisfying size constraints; we allow for different size threshold for different classes. In order to quantify class changes we apply a semantic distance $\delta : \Gamma^2 \to \mathbb{R}_0^+$ with Γ being the set of all land cover classes. With this distance we express the global cost for class change as

$$f_{\text{class}} = \sum_{v \in V} w(v) \cdot \delta(\gamma(v), \gamma'(v)) \,, \tag{1}$$

with V being the set of all areas in the input dataset, $w : V \to \mathbb{R}^+$ their size, $\gamma : V \to \Gamma$ their class before aggregation, and $\gamma' : V \to \Gamma$ their class after aggregation. With the semantic distance δ we can express, for example, that we would rather change a grassland area into farmland (low value for δ) than into settlement (high value for δ). Different authors have proposed methods to derive such measures from given data models (Rodríguez and Egenhofer, 2004; Yaolin et al., 2002).

Additionally, our method allows different compactness measures to be applied. We proved that aggregating areas into contiguous regions while ensuring size constraints and spending a minimum cost for class change is NP-hard (Haunert and Wolff, 2006), which means that we cannot hope to find an efficient and exact algorithm. Therefore, we solved the problem by mixed-integer programming and specialised heuristics. In particular, we introduced a heuristic

(a) original dataset S_{highLoD}

(b) result of collapse by skeletons S_{coll}

(c) result of aggregation S_{aggr}

(d) result of line simplification S_{lowLoD}

Figure 3: Applied generalisation work flow.

that allows arbitrarily large datasets to be decomposed into multiple instances of manageable size (Haunert, 2007). We tested this method on a dataset of ATKIS DLM 50 having an extent of 22 km × 22 km. This corresponds to a complete sheet of a map at scale 1:50 000. The processing took 82 minutes on a Linux server with 4 GB RAM and a 2.2 GHz AMD-CPU. The sample in Fig. 3 was taken from this dataset. Though the settlements in Fig. 3(b) do not have sufficient size for the target LoD, the method creates a settlement of sufficient size by including adjacent forest areas (Fig. 3(c)); this leads to a solution of little class changes and compact shapes. 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%. In summary, with the developed heuristic our approach yields high-quality results in modest time.

In order to avoid cluttered maps at small scale, we apply a line simplification algorithm of de Berg et al. (1998) that defines the simplified line by a subsequence of the original line vertices (Fig. 3(d)). The method ensures the bandwidth criterion, that is, for each vertex of the original line the distance to the simplified line does not exceed a user-defined tolerance. Furthermore, the method ensures the topological correctness of the planar partition. Subject to these constraints the number of vertices in the simplified line is minimised.

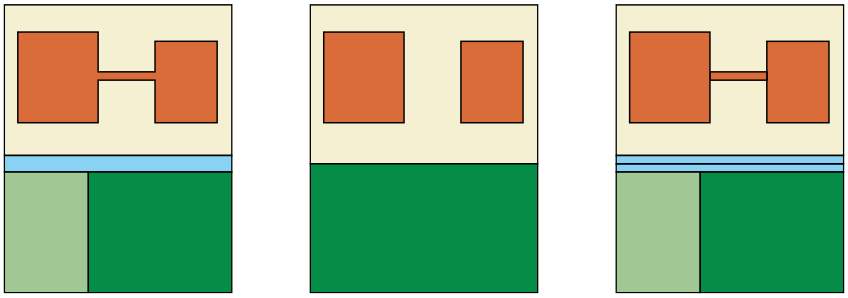
3.2. Generalisation method for intermediate LoDs

In order to define datasets at intermediate LoDs, we aim to find a gradual transformation of the most detailed dataset (S_{highLoD}) into the least detailed dataset (S_{lowLoD}) generated in the first stage (Sect. 3.1). We first discuss a prerequisite for our method and how to ensure it (Sect. 3.2.1). In Sect. 3.2.2 we present an algorithm that defines a sequence of merge operations, in order to gradually transform S_{highLoD} into S_{aggr}, that is, the generalisation result before line simplification. In Sect. 3.2.3 we introduce an improvement of this method that also allows intermediate levels of line simplification to be found and we present results that have been obtained with the improved method.

3.2.1. Defining a dataset containing all geometries

Our intention is to define the transformation between S_{highLoD} and S_{aggr} as a sequence of basic merge operations. However, if we only allow the areas of S_{highLoD} to be merged with each other, we cannot reach S_{aggr}. This is because we applied a collapse operator in our generalisation work flow: Some polygons were decomposed into multiple parts, different parts were potentially merged with different neighbours. Therefore, S_{aggr} contains boundaries that were not present in S_{highLoD}. In order to ensure that we can reach S_{aggr} by applying a sequence of merge operations, we add the additional boundaries of S_{aggr} to S_{highLoD}. In other words, we perform a map overlay between both datasets yielding dataset S_{overlay} that contains all geometries. Each area of S_{overlay} corresponds to exactly one area of S_{aggr}. Figure 4 illustrates this approach. Now we can define a gradual transformation of S_{overlay} into S_{aggr} based on merge operations.

For visualisation purposes, we could hide the additional boundaries until their incident areas become merged with other areas, that is, at high LoDs these



(a) dataset S_{highLoD} (b) dataset S_{aggr} (c) dataset S_{overlay}

Figure 4: Map overlay of most detailed dataset S_{highLoD} and dataset before line simplification S_{aggr}. Result S_{overlay} can be transformed into S_{aggr} using merge operations only.

boundaries are invisible to users. With this approach we can indeed transform S_{highLoD} gradually into S_{aggr}.

3.2.2. Defining a sequence of merge operations

We define the gradual transformation of S_{overlay} into S_{aggr} with a modified version of the iterative algorithm that we previously used to set up the tGAP structure (van Oosterom, 1994, 2005). In each iteration of the original algorithm the least important area is merged with its most compatible neighbour. We also follow this idea in our new approach. However, we introduce requirements in order to force the algorithm to terminate with the given dataset S_{aggr}. Before defining these requirements in detail, we specify our definitions of importance and compatibility.

We denote the importance by the function $I : V' \to \mathbb{R}_0^+$ and the compatibility of areas by the function $C : V' \times V' \to \mathbb{R}_0^+$, where V' is the set of input areas V extended by the areas created from merging subsets of V. These functions are specified as follows:

- The importance $I(v)$ of area $v \in V'$ is equal to its size $w(v)$ multiplied with a class-dependent weight factor $\eta : \Gamma \to [0, 1]$, that is, $I(v) = w(v) \cdot \eta(\gamma(v))$. In our tests we applied a constant weight 1 for all classes.
- The compatibility of area $u \in V'$ to area $v \in V'$ is given by $C(u, v) = \lambda(u, v) \cdot (1 - \delta'(\gamma(u), \gamma(v)))$ with $\lambda : V' \times V' \to \mathbb{R}_0^+$ being the length of the common boundary between two areas, and δ' being the normalised semantic distance between classes, $\delta : \Gamma \times \Gamma \to \mathbb{R}_0^+$ from Sect. 3.1.

In order to formulate the requirements that guarantee to end up with S_{aggr}, we need to introduce two definitions:

- We say that an area is *active*, if it still has a neighbour that is contained in the same composite region.
- For each composite region we define one area of unchanged class as *centre*.

The region centres result as a by-product of our optimisation method (Haunert and Wolff, 2006). Alternatively, we could define the largest area of unchanged class in each region as centre. With these definitions the requirements can be defined as follows:

- The merging process must not be terminated, if there still is an active area.
- An area can only be merged with neighbours in the same composite region, that is, other neighbours are not taken into consideration when selecting the most compatible neighbour.
- A new area resulting from a merge of a centre with another area receives the class of the centre; it becomes the new centre of the region.

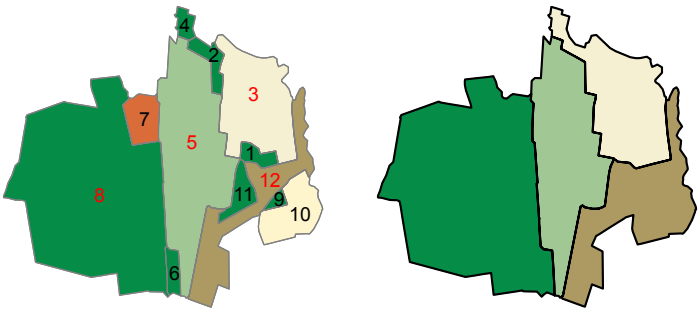
These requirements suffice to guarantee that the iterative algorithm terminates with S_{aggr} when applied to S_{overlay}. Algorithm 1 specifies the approach. In Line 2 the least important active area of the dataset is selected. Line 4 selects its most compatible neighbour from the same composite region. Lines 6 and 9 define the merge operations for the cases that a centre is involved or not, respectively.

Algorithm 1 Generation of intermediate LoDs

```
1:  while there is an active area do
2:    a ← active area with minimum importance I
3:    N ← neighbours of a in the same composite region
4:    b ← b′ ∈ N with highest compatibility C(a, b′)
5:    if a or b is a centre then
6:          merge a and b, new area receives class of centre
7:          define new area as region centre
8:    else
9:          merge a and b, new area receives class of b
10:   end if
11:  end while
```

Figure 5 illustrates the input that we require for Algorithm 1: Each region in the generalised dataset can be obtained by merging a set of areas in the detailed dataset. Furthermore, each region contains a centre. Note that, according to our definition of an active area, the area with ID 5 is the only one that is initially not active: There is no other area contained in the same composite region.

Figure 6 illustrates the stepwise generalisation between the datasets shown in Fig. 5. Each step is labelled with the number of iterations of Algorithm 1 that have been performed. For each step, a red arrow is drawn from the least important area to its most compatible neighbour. In the next iteration, both areas become merged. The process continues until all areas are merged into the



(a) detailed dataset (b) generalised dataset

Figure 5: Samples from two datasets that together can be used as input for Algorithm 1: detailed dataset (a) can be transformed into generalised dataset (b) applying merge operations only. Areas of detailed dataset are labelled with their Ids, region centres are shown by red labels.

composite regions (Step 8). Note that the result is the same as the dataset in Fig. 5(b).

3.2.3. Defining intermediate levels of line simplification

It remains to define intermediate degrees of simplification for lines. We denote a line in dataset S_{overlay} by l_1, its vertices by $\mathcal{V}_1$, and the simplified line in dataset S_{lowLoD} by l_2 with vertices $\mathcal{V}_2 \subseteq \mathcal{V}_1$. 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 $\mathcal{V}$ such that $\mathcal{V}_1 \supseteq \mathcal{V} \supseteq \mathcal{V}_2$, thus an intermediate representation that allows for a refinement by adding vertices when zooming from lowest to highest LoD.

Using Algorithm 1 and the explained procedure for intermediate line simplification levels, we obtain intermediate representations as shown in Fig. 7. The sequence only implies small changes in each step and terminates with the well-generalised dataset obtained from our optimisation method. The constrained tGAP structure is used to record the sequence of changes performed by the iterative generalisation algorithm. The next section presents the components of this structure, how it is set up with the generalisation algorithm, and how it is used for progressive transfer.

4. Constrained tGAP structure and progressive transfer

The generalisation process is performed off-line, its results are stored in the constrained tGAP structure on the server side, which makes it available to clients via an application. For a request sent by a client, the application reads the structure, compiles the response in an ordered manner, and sends data to the client employing the order. Sections 4.1 and 4.2 provide the concepts of our constrained tGAP structure, its components, how it is filled from the

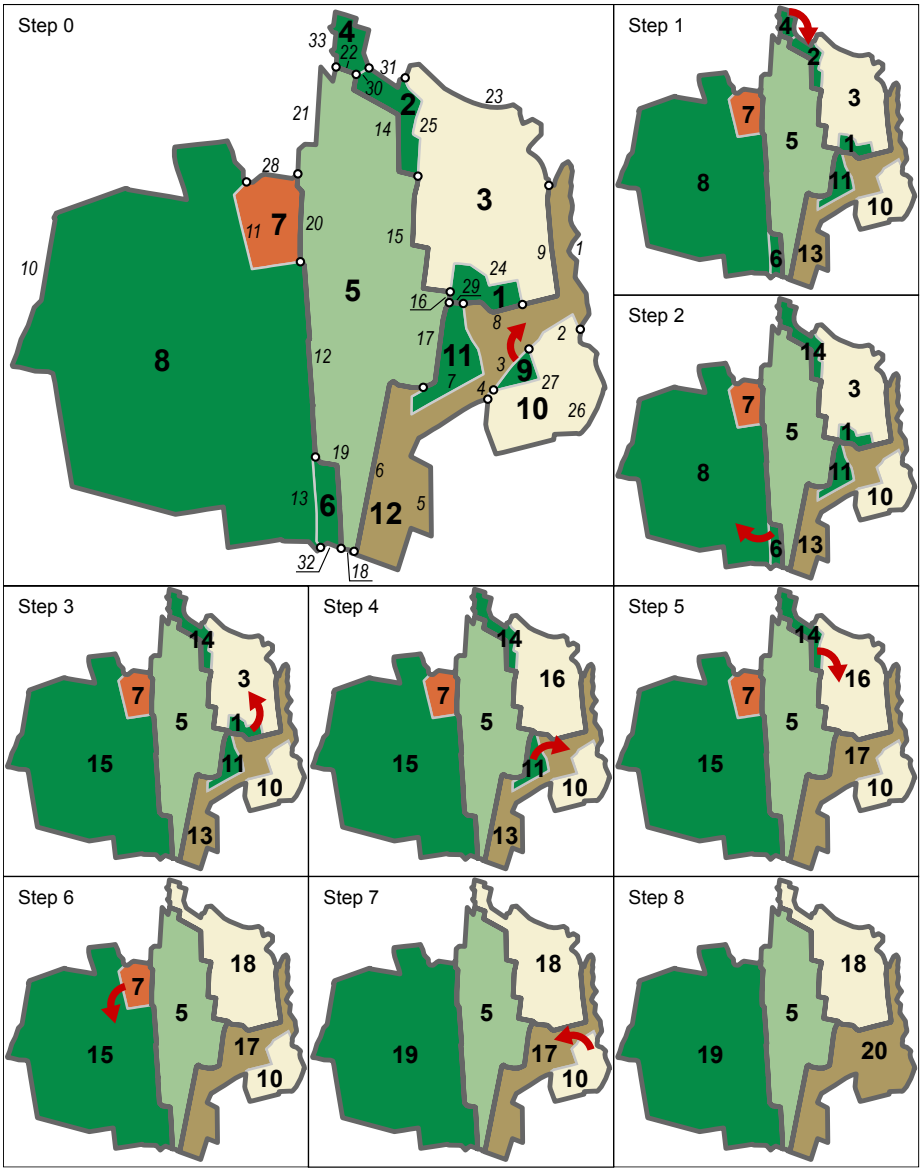


Figure 6: Steps of iterative generalisation with Algorithm 1 for datasets of Fig. 5. Areas are labelled with their Ids, Ids of boundary lines are given for Step 0.

Figure 7: Two examples processed with Algorithm 1. From left to right: most detailed dataset with additional boundaries S_{overlay}, found intermediate representations, and dataset generalised by optimisation S_{lowLoD}.

generalisation process, and how to reconstruct an LoD representation from the structure. Section 4.3 provides the Oracle tables that store the constrained tGAP information. Finally, Sect. 4.4 discusses how to process requests of clients, that is, how to collect data from the Oracle tables and to progressively transfer them.

4.1. Filling the structure with generalisation results

The constrained tGAP structure builds up on the tGAP structure of van Oosterom (2005): it contains additional information related to constraints, that is the composite regions. It uses a topological model for storing data, which guarantees non-redundant geometry and topological consistency for any LoD. The employed topological model consists of faces (that is areas), edges, and nodes. Each edge holds references to its left and right face, as well as to its start and end node. Geometry is stored for edges and nodes, whereas the geometry of a face is constructed by a topology builder algorithm that collects edges referring to it as left or right face.

The constrained tGAP structure stores the results of the generalisation algorithm (Sect. 3.2) in a face hierarchy and an edge hierarchy. Figure 8 shows the face hierarchy created for the sample in Fig. 6; iteration steps are shown to the right. A node in the hierarchy represents an original area (from S_{overlay}) or an area created during generalisation. Leaf nodes are the original areas; they are valid starting from step 0. In every step two areas are merged into a new area: the step ends the validity of the two merged areas; the new area starts being valid from this step. Constraints are at the top of the hierarchy; they

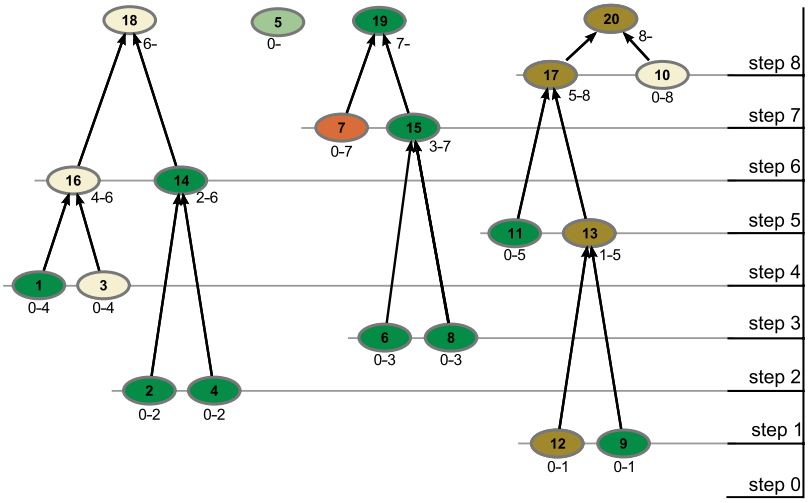


Figure 8: Face hierarchy for generalisation process of Figure 6. Each node corresponds to an area; labels show area Ids and validity ranges (below). Arrows depict merging of two areas into parent area.

continue to be valid after the last step. Each area is associated with its validity range in Fig. 8.

The edge hierarchy holds information about changes happening to edges as the result of area merging. Figure 9 shows the edge hierarchy for the sample in Fig. 6. An edge disappears if it is part of the common boundary of the two merged areas, for example, edge 3 in step 1. These edges are drawn in yellow colour in Fig. 9. The other edges from the boundary of the two merged areas continue existing, but the reference to the left or right face changes, for example, edges $1, 2, 4, 5, 6, 7, 8, 9$, and 27 in step 1. Vertical arrows connecting consecutive versions of an edge represent the range in which an edge is unchanged, i.e., its (references to the) left and right face have not changed.

The constrained tGAP structure stores geometry only for edges and nodes at the highest LoD, S_{overlay}. Areas are constructed following edge references stored in the edge hierarchy. The tGAP generalisation (van Oosterom, 2005; Meijers, 2006) performs edge merging after an area merge, for example, after merging areas 6 and 8 in step 2, edge 12 is merged with edge 19, and edge 10 is merged with 32. For simplicity we skipped the edge merging in our algorithm. The tGAP structure uses BLG (Binary Line Generalisation) trees; for each edge one tree that stores the result of the Douglas-Peucker algorithm (Douglas and Peucker, 1973) for line simplification. Here we chose for on-line simplification and use an optimisation algorithm from de Berg et al. (1998).

4.2. Constructing an LoD representation

We use the term LoD equivalently with map scale. To construct an LoD representation, we first define a correspondence between a map scale and an

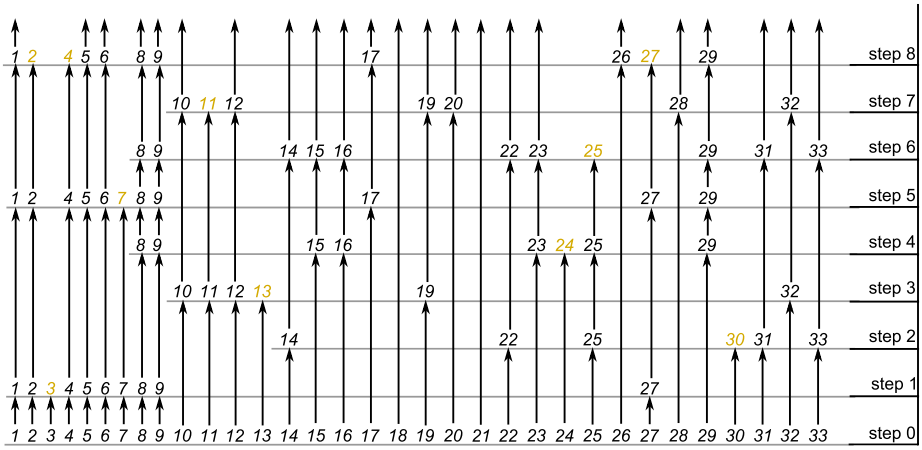
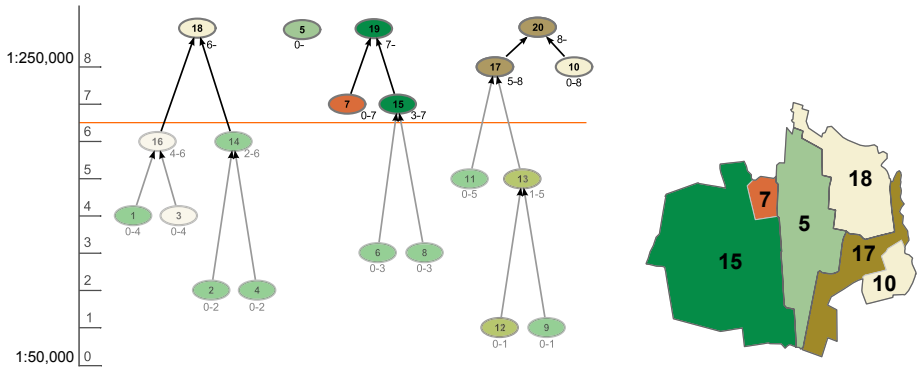


Figure 9: Edge hierarchy for generalisation process of Figure 6.

iteration step in the constrained tGAP. The iterative merging of tGAP allows us to build a relation between iteration steps and number of valid areas in each step. In every step two areas are merged to create a new area; two areas are removed and one is added to the valid areas, thus the number of valid areas decreases by one in each step. This implies the relation $\text{no\_area}(i) = \text{no\_area}(0) - i$, where $\text{no\_area}(i)$ is the number of areas in step i. The relation between scale and the number of map objects often follows certain rules, for example, equation 3 of Töpfer and Pillewizer (1966) states that the ratio between the number of map objects is equal to the ratio between scales. Put differently, this principle states that the ratio between the number of map objects and the scale is constant (assuming the same map extent). We cannot apply the equation directly, as we are given two datasets, S_{overlay} and S_{lowLoD}, of scales 1:50 000 and 1:250 000, and known number of map objects, $\text{no\_area}_{S_{\text{overlay}}}$ and $\text{no\_area}_{S_{\text{lowLoD}}}$, respectively. Though we obtained similar ratios for our datasets, they are not exactly the same. Instead, we consider the ratio between map objects to be proportional with the ratio between scales, which is translated to a linear relation between iteration step i and *scale*:

$$i = \frac{(\text{scale}_{S_{\text{overlay}}} - \mathit{scale}) \cdot (\text{no\_area}_{S_{\text{overlay}}} - \text{no\_area}_{S_{\text{lowLoD}}})}{\text{scale}_{S_{\text{overlay}}} - \text{scale}_{S_{\text{lowLoD}}}} \tag{2}$$

For a given map scale, we calculate the corresponding iteration step i, and search in the constrained tGAP for the valid areas at iteration i. These are the areas that include i in their validity range. Figure 10 illustrates the valid areas of our sample dataset (Fig. 6) at scale 1:125 000, which corresponds to iteration step 6, as calculated from equation 2. The valid areas are the leaf nodes after cutting the face hierarchy at level 6. Following the hierarchy, from the top S_{lowLoD} that is a planar partition, it can be seen that leaf nodes after a cutting form again a planar partition.



(a) select in hierarchy for step 6 (b) LoD representation

Figure 10: Reconstructing a representation of certain LoD by selection in face hierarchy: (a) selecting areas in face hierarchy for scale 1:125 000, corresponding to iteration 6; (b) valid areas at iteration 6. These are leaf nodes after cutting face hierarchy at level 6.

The set of valid edge versions is found similarly from the edge hierarchy. They are the versions which include the calculated iteration i in their range. In the edge hierarchy they are the leaves that remain after a cut at level i. These edges form the boundary of valid faces at iteration i.

4.3. Oracle Spatial tables storing constrained tGAP information

The information of the constrained tGAP structure is stored in Oracle Spatial 10g. Oracle tables and their relationships are shown in Fig. 11. Arrows associating tables show foreign key relationships; cardinalities are shown when different from 1. Primary keys and foreign keys are shown by symbols PK, FK, and pfK for a foreign key that is part of a primary key.

Table Face holds information about areas: an identifier, the class to which it belongs, the composite region to which it is constrained, an attribute with value 1 or 0 that defines whether the area is a centre, the area size, and the validity range as $[\text{imp\_low}, \text{imp\_high})$. Table ClassInfo stores information about classes: code as referred in Face table, name and description, as well as class weight. Table ClassCompatibility stores the compatibility values as cost of changing from the from_class to the to_class. Information about edges is split in two tables: EdgeGeo that stores an identifier, the geometry, its length, and references to start and end node; EdgeLOD that stores references to left, and right area as they change during the generalisation (while the geometry remains the same), and the corresponding validity range $[\text{imp\_low}, \text{imp\_high})$; the combination edge_id, imp_low is unique and it is the primary key of the table. The Node table stores an identifier and the geometry for each node.

The constrained tGAP algorithm reads and writes data from Oracle tables. The algorithm starts with the dataset S_{overlay}, which is stored in the Oracle tables in advance. At each iteration it works with the set of active areas. Changes

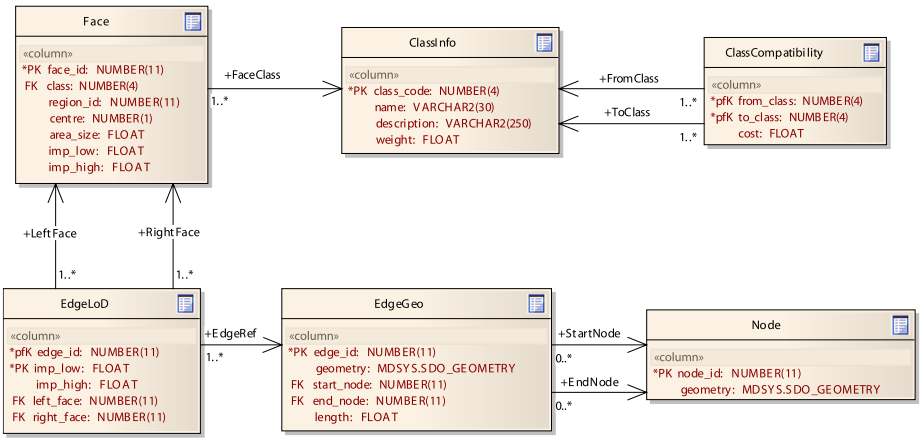


Figure 11: UML diagram of tables and relationships that store constrained tGAP information in Oracle Spatial. PK shows a primary key, FK shows a foreign key, and pfK shows a foreign key that is part of a primary key.

in the face and edge hierarchy performed at each iteration are stored in Face and EdgeLOD tables.

4.4. Progressive transfer in a client-server environment

On a client request for a certain map scale, the server collects information for the corresponding iteration from the constrained tGAP tables. In order to reduce the amount of information transmitted over the network, the server sends only edge geometry and topology information, together with class information for areas. This information is processed by the client prior to visualisation: line simplification is performed; geometry of areas is built from the topology information; areas can then be visualised with their class information. A client application can be simple (stateless), i.e., for any change in scale it requests new data from the server. A more sophisticated client is aware of the tGAP structure. It starts displaying the most coarse representation once the top of the tGAP structure has been received. This initial display is gradually refined as long as more detailed data are received, until the highest LoD data have been received. This client builds its own (partial) copy of the tGAP structure, which can be used afterwards for displaying maps at different scales (supporting local smooth zooming). The next two sections present the server-client communication for these two types of clients.

4.4.1. Simple client

The client sends a request to the server for a map scale defined by a user. The requested scale is transformed on the server side into the corresponding iteration step in the constrained tGAP structure. Data for the valid areas at this step are collected and sent to the client following an importance order.

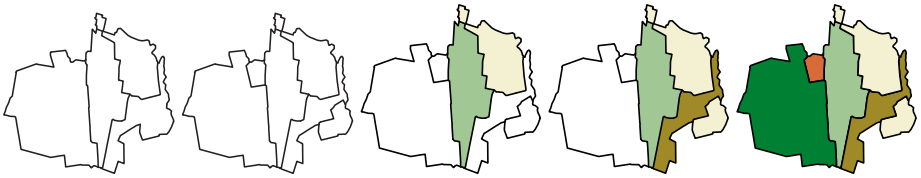


Figure 12: Visualising sample data at scale 1:125 000 (iteration 6) on importance order: (a) valid edges of `imp_high` > 8; (b) valid edges of `imp_high` > 6; (c)–(e) valid areas of `imp_high` greater than 8–6, respectively. Edges are drawn first in the importance order; when all edges are received areas are built and visualised based on class information.

Queries that collect data from the Oracle tables for a requested scale, e.g., scale 1:125 000, corresponding to iteration step 6 in our example, are:

```
select g.edge_id, g.start_node, g.end_node, g.geometry,
       t.left_face, t.right_face
from EdgeLoD t join EdgeGeo g on t.edge_id = g.edge_id
where t.imp_low <= 6 and t.imp_high > 6
order by t.imp_high desc;

select face_id, class
from Face
where imp_low <= 6 and imp_high > 6
order by imp_high desc;
```

The first query collects edge information, the second collects area information. The `where` condition assures the selection of edges and areas that should appear at the requested scale, i.e., valid edges and areas at the corresponding iteration. Ordering by `imp_high` allows the server to send edges according to their importance, that is, edges that are visible at smaller scales are sent first. Once the full set of edges is received, the client builds the geometry of areas, and colours them based on class information.

Not considered in our examples, but important in practice, is that queries should also consider the spatial extent.

Figure 12 illustrates the order of visualising our sample data (Fig. 5(a)) at scale 1:125 000, which corresponds to iteration 6 in the constrained tGAP. First, valid edges at iteration 6 are shown in order of importance; most important edges reside higher in the hierarchy (Fig. 9). Figures 12(a) and 12(b) illustrate the order. Once all valid edges are received (Fig. 12(b)), the client builds the geometry of valid areas for iteration 6, areas 5, 7, 10, 15, 17, and 18 (Fig. 10(b)). More important areas are placed higher in the face hierarchy (Fig. 8 and Fig. 10(a)). Figures 12(c), 12(d), and 12(e) visualise areas following their importance, that is the `imp_high` value.

4.4.2. Smarter client

A smarter (and heavier) client can use the data sent from the server to build its own copy of the tGAP structure. For a client request at a given scale, the server sends the data from the most coarse scale (lowest LoD) up to the requested scale. These data then allow for progressive transfer from server to client and smooth zooming on the client side. For this, the hierarchies are traversed in the opposite order of their creation. For example, on a client request for a map scale up to 1:125 000 of our sample, the server collects data for areas at the lowest LoD (top of the hierarchy of Fig. 8), and follows with data for every split going down in the face hierarchy, until iteration 6 (that corresponds to scale 1:125 000) is reached.

We create a view that records the full validity range for an edge geometry table, and use this view for easier selection in the subsequent queries:

```
create view EdgeLoD_range(edge_id, imp_low, imp_high) as
    select edge_id, min(imp_low), max(imp_high)
    from EdgeLoD
    group by edge_id;
```

Three separate queries run on the server side, collecting geometry information of edges, topology information for edges, and area information, all in increasing level of detail. The SQL queries collecting data for our sample dataset from the top of the hierarchy to iteration 6 (used by the illustration of Fig. 13) are given below:

```
select g.edge_id, geometry, start_node, end_node, imp_high
from EdgeGeo g join EdgeLoD_range l on g.edge_id = l.edge_id
where imp_high > 6
order by imp_high desc, edge_id;

select edge_id, left_face, right_face, imp_low, imp_high
from EdgeLoD
where imp_high > 6
order by imp_high desc, edge_id;

select face_id, class, imp_low, imp_high
from Face
where imp_high > 6
order by imp_high desc;
```

Once a specific scale has been received, a client can make a refinement request for a larger scale. The open range condition in the `where` clause of the above SQL queries is replaced with a closed range condition; for example, for the scale range 1:125 000 – 1:62 500, which corresponds to the iterations range 6–2, the condition is modified to `imp_high <= 6 and imp_high > 2`.

In both situations, an initial and a refinement request, the server collects with three separate 'parallel' queries the geometry information of edges, their

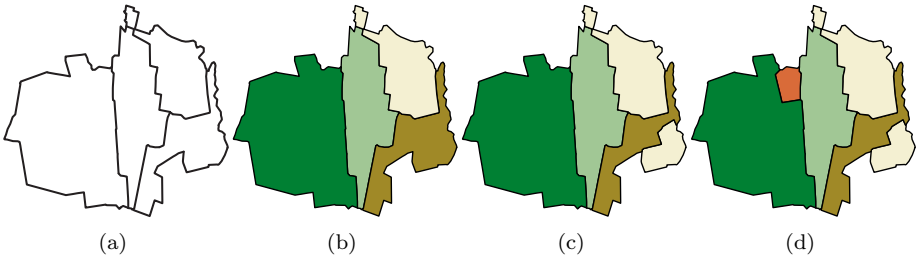


Figure 13: Visualising data from lowest LoD to a requested LoD, adding refinement following opposite order of iterative generalisation (Fig. 6): (a) edges of lowest LoD; (b) areas of lowest LoD; (c) refined by first split (iteration 8); (c) further refined by second split (iteration 7).

topology information, and the area information. Three cursors in a program on the server side iterate over the output of these SQL queries in Oracle tables. The ordering on `imp_high` allows for grouping of three different record types related to one iteration step (representing the next refinement). One package contains the used edges and the areas involved in a step of the tGAP structure creation: two areas are merged and at least one edge is removed. The group of records related to one step, is 'packaged' together and written to an XML (or GML) document for immediate data transfer to the client. As soon as the first package is complete and received on the client side, the client can use these data for the most coarse representation. The server continues writing and sending the document until the last package has been compiled and sent. The Transmission Control Protocol/Internet Protocol (TCP/IP) guarantees that the document is correctly received. The result is a data stream from server to client. The client keeps on receiving data in the right order and can refine the representation package by package. The received information is stored in a local partial copy of the tGAP structure on the client side.

Figure 13 illustrates a client visualisation from this progressive transmission of data: Fig. 13(a) shows valid edges after the last iteration, step 8; Fig. 13(b) shows valid areas after iteration step 8, areas 5, 18, 19, and 20; Fig. 13(c) replaces area 20 that was merged at step 8 with its children, 17 and 10; Fig. 13(c) follows by replacing the area 19 that was merged at step 7 with its children, 7 and 15. The information displayed on the client side is refined after reception of each package, until the required level of detail is reached.

5. Conclusion and future work

5.1. Future work

Our method is based on the assumption that the aggregation is the dominating relationship between areas of two given datasets. Additional lines resulting from the collapse operation are simply included in the original dataset (resulting in S_{overlay}). 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 lines and areas. Instead of having the collapse operation (and other operations) only available as preprocessing operation, it might also be fully integrated in the (constrained) tGAP structure. The effect of including the area-to-line collapse function is that the tGAP face hierarchy (now tree for tGAP, and acyclic graph for the constrained tGAP) will become a tGAP face-DAG (directed acyclic graph) as the collapsed area will be partitioned and assigned to multiple parents (Ai and van Oosterom, 2002). However, this will happen at most once for every area and does fit well in the proposed table structure. The area-to-line collapse may be especially useful for infrastructure at large scale, such as road and water networks, which are often represented at the medium scales by line objects. Care has to be taken that the (road or water) network topology is maintained and the line representations are connected to each other.

Until now we do not have empirical results concerning the inclusion of additional operators. For the future, we plan to perform tests on a dataset of realistic size. We also plan to test our method with two different settings for the line simplification. In our current implementation edges are never merged, but as described in earlier papers it is possible to join BLG-trees of two edges (without geometric redundancy). Alternatively, it is possible to compute geometry for a merged edge representation and store the result in the tGAP structure. This would introduce some geometric redundancy but would have less overhead or references as in the situation of the joined BLG-tree solution. In the case of joined BLG-trees, sending edges to the client would be done by progressively sending the (stored and merged) BLG-trees to the client. In case of the new geometric edges, this is send to the client (more geometry but less structure/references) and if needed the client could also apply further line simplification on-the-fly.

The list of future work further contains a collection of old and new ideas, yet to be included into the structure in order to further refine the quality and performance of the (constrained) tGAP structure:

- In our current prototype we did not fine-tune the class weights and the class compatibility matrix. However this does influence the tGAP creation as described in earlier publications; (van Putten and van Oosterom, 1998, e.g.,).

- Only the importance was included in the current selection conditions when using the tGAP structure. However, also the spatial extend should be included. In case of the simple client this is straightforward as just one slice of importance at a certain location is needed. In case of the smarter client (smooth zoom) it is more complicated. Assume that the user wants to zoom in, the spatial extend rectangle would become smaller while the scale becomes larger (i.e., importance gets lower). Because also everything between the current larger area (at small scale) and the new smaller area (at larger scale) is needed, this is a kind of truncated pyramid selection (in 3D with xy being the spatial extend and z being the importance dimension).

- As a user may pan and zoom in and zoom out, the client receives more and more data, which can be maintained in a local copy of the tGAP-structure. This reduces the amount of data to be requested from the server for the subsequent queries. However, this cache management is non trivial due to the truncated pyramid shapes of the selections. Therefore alternative solutions are being investigated which use more regular blocks of selected data. These allow for easier administration of the cache content and easier formulation of queries.

- When line and point features are to be included in the tGAP structure, either as input data or created during the processing (area-to-line or area-to-point collapse) then also support is needed for these objects; for example also edges can now get an object classification code (similar to the current faces/areas). Also links between the different representations of the same object should be maintained; e.g., between an area and line representation of the same road.

- Instead of first computing an optimised generalised representation and then computing the tGAP structure to gradually change from the original large scale data to the generalised representation, it would also be possible to use an external generalised representation (of another source) as target. In this case more matching problems occur and have to be solved when linking the large objects to the smaller scale regions on another independent source (Hofman et al., 2008).

- Due to changes in left and right references there are many more instances of EdgeLoD than there are instances of EdgeGeo. If the left and right references are omitted, then EdgeLoD and EdgeGeo can be replaced with one Edge table (more compact storage and less data to be transferred to the client). After forming the areas on the client side, the client has to match the Face information with the formed areas; one option for this is to add a centroid to the Face table. This requires more topology processing by the client, but less data is stored and transferred, so it is expected that the overall performance will be improved.

- It is nice to have a data structure that supports progressive transfer and smooth zoom principles, but without a good client it is very difficult to fully exploit this. Therefore future research will also include the design and development of a client supporting smooth zoom.

5.2. 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 variable-scale structure of high quality. The unconstrained tGAP structure may result in medium and small-scale representations of less quality. Using constraints, either computed (see Section 3.1) or from another medium/small-scale source, will guarantee that the quality at the constrained 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:

- 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)
- it does support progressive transfer of vector data to be used in smooth zoom functionality on the client side.

Acknowledgements

This paper presents results of the project "Usable (and well scaled) mobile maps for consumers", which was part of the Bsik Space for Geo-information (RGI) programme (see `http://www.rgi-otb.nl/uwsm2`).

References

Ai, T., Li, Z., Liu, Y., 2005. Progressive transmission of vector data based on changes accumulation model. In: Fisher, P. (Ed.), Developments in Spatial Data Handling: 11th International Symposium on Spatial Data Handling. Springer Verlag, Berlin, Germany, pp. 85–96.

Ai, T., van Oosterom, P., 2002. GAP-Tree extensions based on skeletons. In: Richardson, D., van Oosterom, P. (Eds.), Advances in Spatial Data Handling: 10th International Symposium on Spatial Data Handling. Springer Verlag, Berlin, Germany, pp. 501–513.

Bader, M., Weibel, R., 1997. Detecting and resolving size and proximity conflicts in the generalization of polygonal maps. In: Proceedings 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: Proceedings 20th International Cartographic Conference (ICC'01), 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, Harlow, UK, Ch. 7, pp. 121–149.

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

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

Brenner, C., Sester, M., 2005. Continuous generalization for small mobile displays. In: Agouris, P., Croituru, A. (Eds.), Next Generation Geospatial Information. Taylor & Francis, London, UK, pp. 33–42.

Buttenfield, B. P., 2002. Transmitting vector geospatial data across the internet. In: Egenhofer, M. J., Mark, D. M. (Eds.), Proceedings 2nd International Conference on Geographic Information Science (GIScience'02), Boulder, CO, USA. Vol. 2478 of Lecture Notes In Computer Science. Springer Verlag, Berlin, Germany, pp. 51–64.

Cecconi, A., 2003. Integration of Cartographic Generalization and Multi-Scale Databases for Enhanced Web Mapping. Dissertation, University of Zurich, Switzerland, 196 pp.

Cheng, T., Li, Z., 2006. Toward quantitative measures for the semantic quality of polygon generalization. Cartographica 41 (2), 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., 2005. Multi-resolution extension for transmission of geodata in a mobile context. Computers & Geosciences 31 (2), 179–188.

Galanda, M., 2003. Automated Polygon Generalization in a Multi Agent System. Dissertation, University of Zurich, Switzerland, 176 pp.

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., 2007. Efficient area aggregation by combination of different techniques. In: Proceedings 10th Workshop of the International Cartographic Association (ICA) Commission on Generalisation and Multiple Representation, Moscow, Russia.
URL `http://aci.ign.fr/BDpubli/moscow2007/Haunert_ICAWorkshop.pdf`

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: Proceedings 14th Association for Computing Machinery International Symposium on Advances in Geographic Information Systems (ACM-GIS'06), Arlington, VA, USA. pp. 75–82.

Hofman, A., Dilo, A., van Oosterom, P., Borkens, N., 2008. Using the constrained tGAP for generalisation of IMGeo to Top10NL model. In: Proceedings 11th Workshop of the International Cartographic Association (ICA) Commission on Generalisation and Multiple Representation, Montpellier, France.
URL `http://aci.ign.fr/montpellier2008/papers/27_Hofman_et_al.pdf`

Jaakkola, O., 1997. Quality and Automatic Generalization of Land Cover Data. Dissertation, University of Helsinki, Finland, 39 pp.

Meijers, M., June 2006. Implementation and testing of variable scale topological data structures. Master's thesis, TU Delft, The Netherlands, 92 pp.

Merrick, D., Nöllenburg, M., Wolff, A., Benkert, M., 2007. Morphing polygonal lines: a step towards continuous generalization. In: Proceedings Geographical Information Science Research UK Conference (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.

Papadimitriou, C. H., Steiglitz, K., 1998. Combinatorial Optimization: Algorithms and Complexity. Dover Publications, Inc., Mineola, NY, USA, 512 pp.

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.

Töpfer, F., Pillewizer, W., 1966. The principles of selection. The Cartographic Journal 3 (1), 10–16.

van Oosterom, P., 1994. Reactive Data Structures for Geographic Information Systems. Oxford University Press, UK, 216 pp.

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 (4), 331–346.

van Putten, J., van Oosterom, P., 1998. New results with generalized area partitionings. In: Proceedings 8th International Symposium on Spatial Data Handling (SDH'98), Vancouver, Canada. pp. 485–495.

van Smaalen, J. W. N., 2003. Automated Aggregation of Geographic Objects. Dissertation, Wageningen University, The Netherlands, 104 pp.

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: Proceedings 8th International Symposium on Spatial Data Handling (SDH'98), Vancouver, Canada. pp. 214–224.

Yang, B., 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 International Society for Photogrammetry and Remote Sensing (ISPRS) Commission IV Symposium on Geospatial Theory, Processing and Applications, Ottawa, Canada. No. XXXIV(Part 4) in International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences.
URL `http://www.isprs.org/commission4/proceedings02/pdfpapers/413.pdf`