3 Vario-scale data structures

The previous chapter presents state-of-the-art in map generalization at NMAs' and continuous generalization. There is a noticeable technological shift towards continuous generalisation which supports interactive map use where users can zoom in, out and navigate more gradual way. Despite some research efforts there is no satisfactory solution yet. Therefore, this chapter introduces the truly smooth vario-scale structure for geographic information where a small step in the scale dimension leads to a small change in representation of geographic features that are represented on the map. With this approach there is no (or minimal) geometric data redundancy and there is no (temporal) delay any more between the availability of data sets at different map scales (as was and is the case with more traditional approaches of multi-scale representations). Moreover, continuous generalisation of real world features is based on the structure that can be used for presenting a smooth zoom action to the user.

More specific, Section 3.1 and 3.2 provide historical overview of the development and the theoretical framework for vario-scale representations: the tGAP-structure (topological Generalized Area Partitioning). Section 3.3 describes the initial effort to generate the better cartographic content; the concept of constraint tGAP. Section 3.4 explains the 3D SSC (Space-Scale Cube) encoding of 2D truly vario-scale data. Section 3.5 shows idea how to combine more level of details in one map. Section 3.6 summarizes the open questions of the vario-scale concept and it indicates research covered in following chapters. Finally, Section 3.7 presents vario-scale data research in parallel to this PhD for progressive data transfer. Then, Section 3.8 summarises the chapter.

Own publications

This chapter is based on the following own publications:

- van Oosterom, P., Meijers, M., Stoter, J., and Šuba, R. (2014). *Abstracting Geographic Information in a Data Rich World: Methodologies and Applications of Map Generalisation*, volume 2014 of *Lecture Notes in Geoinformation and Cartography*, chapter Data Structures for Continuous Generalisation: tGAP and SSC, pages 83–118. Springer International Publishing.
- Šuba, R., Meijers, M., Huang, L., and van Oosterom, P. (2014a). An area merge operation for smooth zooming. In Huerta, J., Schade, S., and Granell, C., editors, *Connecting a Digital Europe Through Location and Place*, Springer Lecture Notes in Geoinformation and Cartography, pages 275–293. Springer International Publishing. ISBN: 978-3-319-03611-3.
- Huang, L., Meijers, M., Šuba, R., and van Oosterom, P. (2016). Engineering web maps with gradual content zoom based on streaming vector data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 114:274 – 293.

§ 3.1 History of development

Data structures for multi-scale databases based on multiple representations attempt to explicitly relate objects at different scale (or resolution) levels, in order to offer consistency during the use of the data. The drawbacks of the multiple representations data structures are that they do store redundant data (same coordinates, originating from the same source) and that they support only a limited number of scale intervals. Most data structures are intended to be used during the pan and zoom (in and out) operations, and in that sense multi-scale data structures are already a serious improvement for interactive use as they do speed-up interaction and give reasonable representations for a given level of detail (scale).

NEED FOR PROGRESSIVE DATA TRANSFER: A drawback of multiple representation data structures is that they are not suitable for progressive data transfer, because each scale interval requires its own (independent) graphic representation be transferred. Good examples of progressive data transfer are raster images, which can be presented relatively quickly in a coarse manner and then refined as the user waits a little longer. These raster structures can be based on simple (raster data pyramid) (Samet, 1984) or more advanced (wavelet compression) principles (Lazaridis and Mehrotra, 2001; Hildebrandt et al., 2010; Rosenbaum and Schumann, 2004). For example, JPEG2000 (wavelet based) allows both compression and progressive data transfer from the server to the end-user. Also, some of the proprietary formats such as ECW from ER Mapper and MrSID from LizardTech are very efficient raster compression formats based on wavelets and offering multi-resolution access suitable for progressive data transfer. Similar effects are more difficult to obtain with vector data and require more advanced data structures, though a number of attempts have been made to develop such structures (Bertolotto and Egenhofer, 2001; Buttenfield, 2002; Jones et al., 2000; Zhou et al., 2004).

MULTI-SCALE / VARIABLE-SCALE VECTOR DATA STRUCTURES FOR LINE FEATURES: For single (line) objects, a number of multi-scale/variable-scale data structures have been proposed: Strip-tree (Ballard, 1981), Multi-Scale Line tree (Jones and Abraham, 1987), Arc-tree (Günther, 1988), and the Binary Line Generalisation tree (BLG-tree) (van Oosterom, 1990). The Strip-tree and the Arc-tree are intended for arbitrary curves, not for simple polylines. The Multi-Scale Line tree is intended for polylines, but it introduces a discrete number of detail levels and it is a multi-way tree, meaning that a node in the tree can have an arbitrary number of children. The BLG-tree is a binary tree for a variable-scale representation of polylines, based on the Douglas and Peucker (1973) line generalisation algorithm but can be combined with several other line generalization algorithms. Note that these line data structures can only be used for spatial organization of single objects and for (indexing, clustering) of multiple objects (as needed by variable-scale or multi-scale map representations), so they only solve part of the generalisation and storage problem.

One of the first multi-scale vector data structures designed to avoid redundancy was the reactive BSP-tree (van Oosterom, 1989), which supports both spatial organization (indexing) and multiple level of details. Its main disadvantage, however, is that it is a static structure. The first dynamic vector data structure supporting spatial organization of all map objects, as well as multiple scales, was the Reactive tree (van Oosterom,

1992, 1994). The Reactive tree is an R-tree (Guttman, 1984) extension with importance levels for objects: more important objects are stored higher in the tree structure, which makes more important object more accessible. This is similar to the reactive BSP-tree, but the dynamic structure of the Reactive tree enables inserts and deletes, functions that the BSP-tree lacks. The BLG-tree and the Reactive tree are eminently capable of supporting variable-scale/multi-scale maps composed of individual polyline or polygon objects.

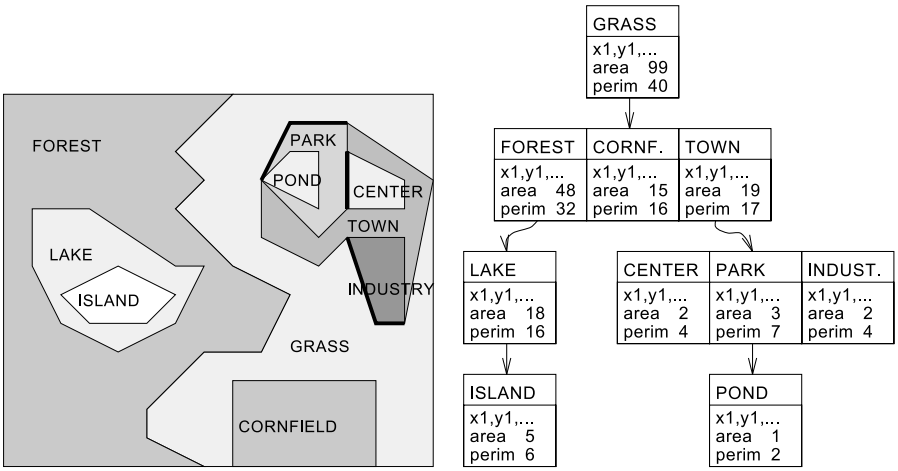
GENERALIZED AREA PARTITIONING: The BLG-tree and Reactive-tree structures are not well suited for an area partitioning, since removal of a polygon results in a gap in the map and independent generalisation of the boundaries of two neighbour areas results in small slivers (overlaps or gaps). Overcoming this deficiency was the motivation behind the development of the GAP tree (van Oosterom, 1995). The BLG-tree, Reactive-tree, and GAP-tree data structures can be used together, while each supports different aspects of the related generalisation process, such as selection and simplification, for an area partitioning (van Oosterom and Schenkelaars, 1995).

Following the conceptualization of the GAP tree, several improvements were published to resolve limitations of the original data structures (van Putten and van Oosterom, 1998; Ai and van Oosterom, 2002; Vermeij et al., 2003). The next section describes the background of the topological GAP tree, which combines the use of the BLG-tree and the Reactive tree and avoids the problems of the original GAP tree – redundant storage and slivers near the boundary of two neighbour areas.

§ 3.2 GAP tree background

The first tree data structure for generalised area partitioning (GAP tree) was proposed by van Oosterom (1995). The idea was based on first drawing the larger and more important polygons (area objects), so as to create a generalised representation. However, one can continue by refining the scene through the additional drawing of the smaller and less important polygons on top of the existing polygons (based on the Painters algorithm; see Figure 3.1. This principle has been applied to the Digital Land Mass System-Digital Feature Analysis Data (DLMS DFAD) data structure (DMA USDMA, 1986), because it already had this type of polygons organization. When tested with the Reactive tree and the BLG-tree, it was possible to zoom in (zoom out) and obtain map representations with more (less) detail of a smaller (larger) region in constant time (see Figure 3.3, left).

COMPUTING THE GAP TREE: If one has a normal area partition (and not DLMS DFAD data) one first has to compute the proper structure. This is driven by two functions. First, the importance function (for example: *Importance(a) = Area(a) * WeightClass (a)*) is used to find the least important feature a based on its size and the relative importance of the class it belongs to. Then the neighbour b is selected based on the highest value of *Collapse(a,b) = Length(a,b) * CompatibleClass(a,b)*, with *Length(a,b)* being the length of the common boundary. Feature a is removed and feature b takes its space on the map. In the GAP tree this is represented by linking feature a as the child of parent b (and enlarging the original feature b). This process is repeated until only one feature is left covering the whole domain, forming the root of the GAP tree. Figure 3.1 gives a schematic representation of such a GAP tree.



(a)The scene (b)The GAP tree

FIGURE 3.1 The original GAP tree (van Oosterom, 1995).

Work by van Smaalen (2003) focuses on finding neighbour patterns, which might in turn be used for setting up an initial *compatibility matrix*. Bregt and Bulens (1996) give area generalisation examples in the domain of soil maps, based on the same principles. Both van Smaalen (2003) and Bregt and Bulens (1996) use an adapted classification for the higher (merged) level of objects, instead of keeping the original classification at all levels of detail; e.g., deciduous forest and coniferous forest objects are aggregated into a new object classified as 'forest' or 'garden', while 'house' and 'parking place' objects form the new object 'lot'. This could also be done in the GAP tree.

IMPLEMENTATIONS AND IMPROVEMENTS OF THE GAP TREE: Though the GAP tree may be computed for a source data set, which has a planar partitioning topology, the GAP tree itself is not a topological structure. Each node in the GAP tree is a polygon, and this introduces some redundancy as parents and child may have some parts of their boundary in common. The first GAP-tree construction based on topologically structured input was implemented by van Putten and van Oosterom (1998) for two real world data sets: Top10vector (1:10,000) and GBKN (1:1,000; Figure 3.3, right). It turned out that finding the proper importance and compatibility functions (which drive the GAP-tree construction) is far from trivial and depends on the purpose of the map. In addition, two improvements were presented in the 1998 paper (at the conceptual level): 1) adding parallel lines to 'linear' area features, and 2) computing a GAP tree for a large seamless data set.

Ai and van Oosterom (2002) presented two other possible improvements to the GAP tree: One improvement was that the least important object should not only be assigned to one neighbour, but subdivided along its skeleton and the different parts assigned to different neighbours/parents (the result is not a tree but a directed acyclic graph: GAP-DAG). The second improvement concerned extending the neighbourhood analysis by considering non-direct (sharing a common edge) neighbour areas as well. Both suggestions are based on an analysis using a Triangular Irregular Network (TIN) structure.

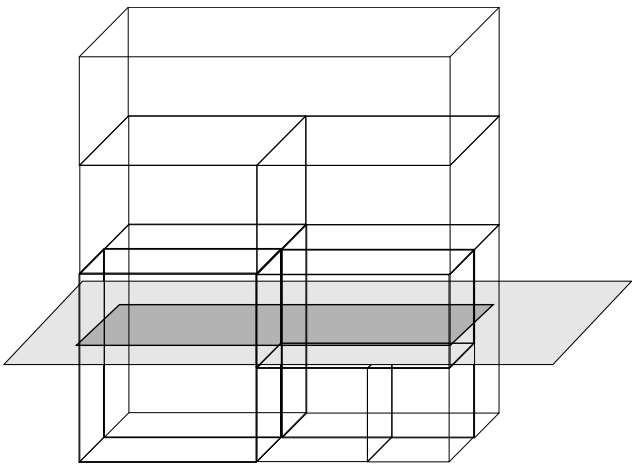


FIGURE 3.2 Importance levels represented by the third dimension (Figure 3 of (Vermeij et al., 2003)). At the most detailed level (bottom) there are several objects, while at the most coarse level (top) there is only one object. The hatched plane represents a requested level of detail, and the intersection with the symbolic 3D volumes then gives the faces.

TOPOLOGICAL VERSION OF THE GAP TREE: All improvements still result in a non-topological GAP structure, which means that it contains redundancy. Vermeij et al. (2003) presented a GAP-tree structure that avoids most redundancy by using a topological structure for the resulting GAP tree, not only for the input: thus the edges and the faces table both have attributes that specify the importance ranges in which a given instance is valid. The 2D geometry of the edges (and faces) is extended by the importance value range (on the z-axis) for which it is valid (see Figure 3.2). One drawback of this approach is that it requires considerable geometric processing at the client side – clipping edges, forming rings, and linking outer and possible inner rings to a face. A second drawback is that there is some redundancy introduced via the edges at the different importance levels: *i. e.* the coordinates of detailed edges are again present in the edge at the higher aggregation level. Finally, van Oosterom (2005) introduced data structure which also avoids the redundant storage of geometry at different levels of detail (in the edges). Figure 3.4 shows the generalisation process for the tGAP structure in which also a simplified version of the edges is available, without explicitly storing them.

MORE COMPACT TGAP STORAGE: The structure stored the node and face data very efficiently. However, there was a lot of redundancy in the way the edges were stored. The edge table for a realistic data set did have up to 15 times more rows than the original edge table (and the theoretic worst case is even $O(n^2)$ with n the number of edges at the largest scale). This was mainly due to the changing references to the left and right faces after merging two neighbour faces. Therefore, Meijers et al. (2009) have proposed the modification of structure which removes this redundancy without loss of functionality.

It was based on two following aspects; First, multiple edge rows in the edge table may merge in one row if 1) they relate to the same edge and 2) only the left/right references were changed (not the edge geometry). This results in no change for the geometry, start and end nodes, and id attributes. The imp_low and imp_high attributes con-

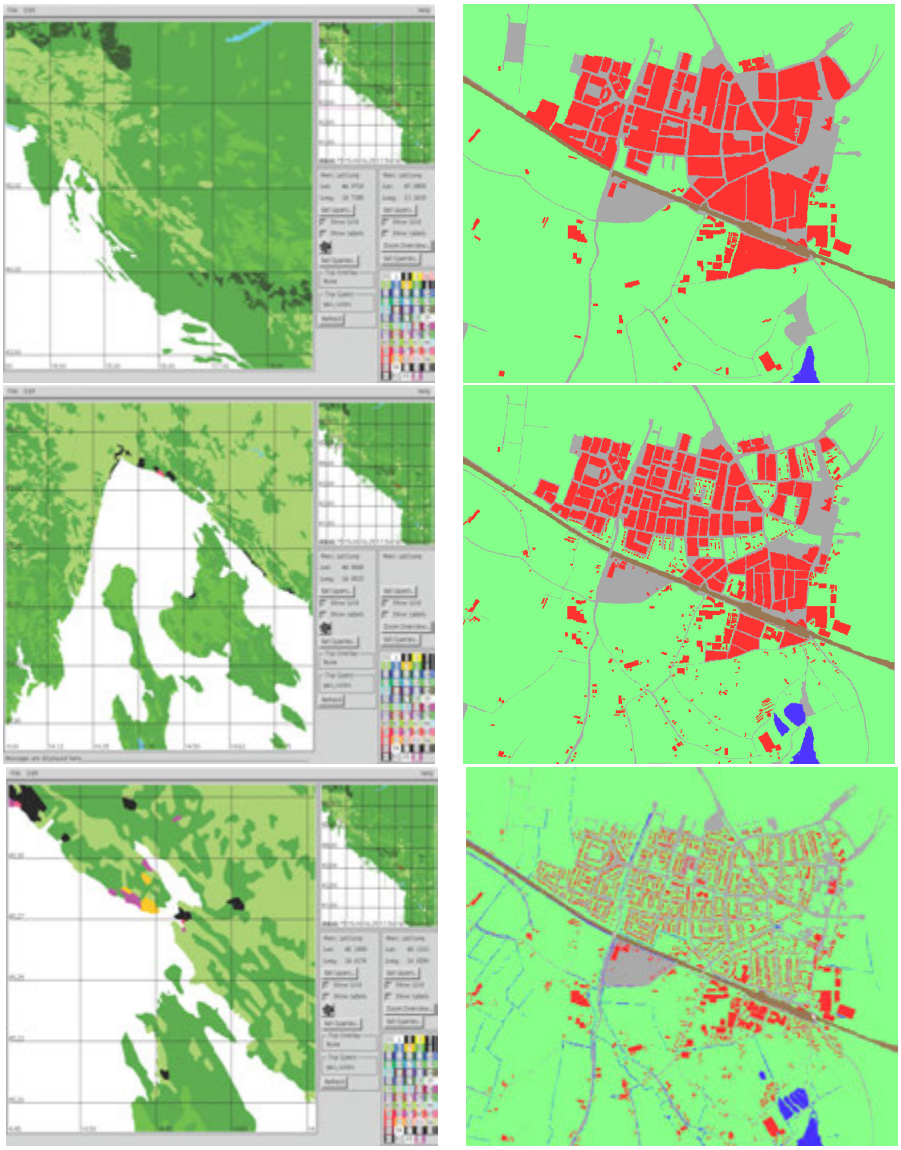


FIGURE 3.3 Left: GAP-tree principle applied to DLMS DFAD (add detail when zooming in). Right: GAP tree applied to large scale topographic data set (shown at same scale). Taken from (van Oosterom, 2005).

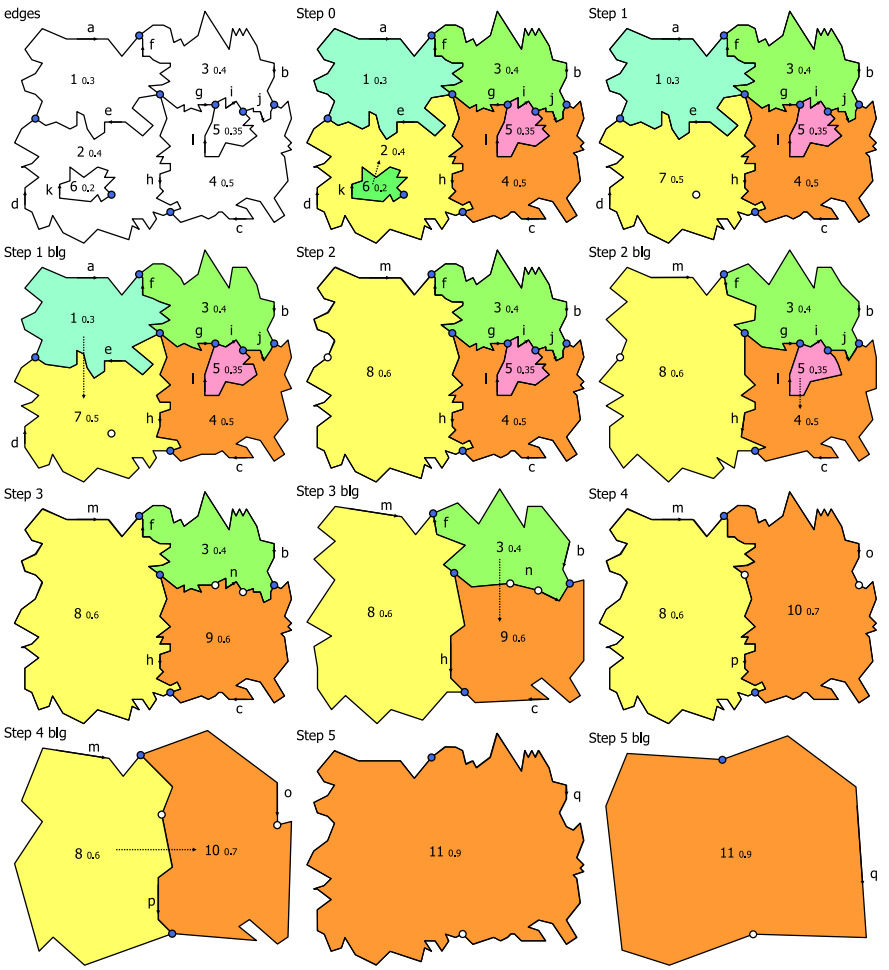


FIGURE 3.4 Generalisation example in five steps, from detailed to course, where merge and simplification operations used. Steps with *blg* in the title shows the effect of simplifying the boundaries via the BLG tree. Note that nodes are depicted in blue and removed nodes are shown for one next step only in white. Taken from (van Oosterom, 2005).

tain the union of all importance ranges of the edge (which are per definition adjacent ranges). Second, the left and right face references should be stored related to the lowest importance range. In these cases the referred face (and the corresponding edge) with the highest imp_low level is used as start in the tGAP face-tree and the tree is traversed upwards until the face identifier at the right imp level is found. The left/right information and the tGAP face-tree can then be exploited to properly identify the areas at a certain importance level (scale).

Then, just storing only rows for edges that are really new (because these edges are merged) saves a lot of storage (rows) (the number of rows reduced by a factor of at least 2 as edges are merged pairwise) (Meijers et al., 2009, p. 14). Saving storage space also implies saving data transfer times in a server-client architecture.

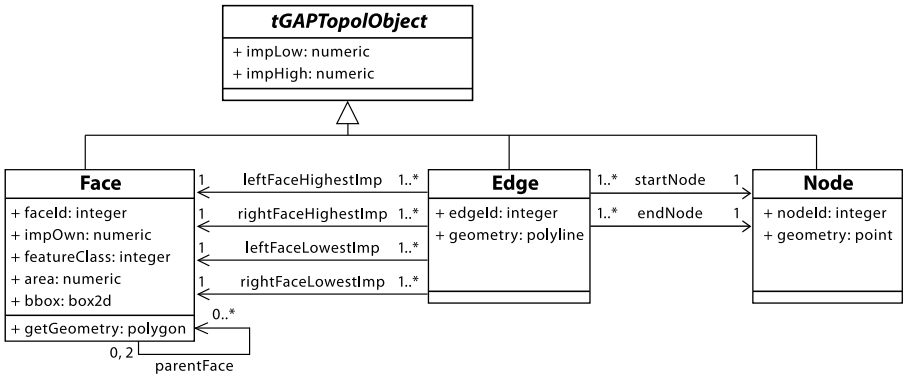
DESCRIPTION OF THE STRUCTURE: All modifications resulted in the tGAP structure captured in Figure 3.5. The figure summarizes the structure in current implementation, its database tables and UML diagram.

The description originates from (Meijers, 2011, p. 157); "Edges table (polylines) is the most important in the database. It contains edges, The imp_low and imp_high attributes define the range of map scales for which a topological entity is valid and has to be shown. Face references that are stored are limited to the neighbours that are adjacent at the start of such a scale range ($left\_face\_lowest\_imp$ and $right\_face\_lowest\_imp$) and which faces are neighbouring at the end of this range ($left\_face\_highest\_imp$ and $right\_face\_highest\_imp$). This way it is prevented that a lot of duplicate edge records have to be stored, while the only thing that changes (due to a generalisation operation) is a neighbouring face. Note that the two extra face pointers (at the high end of the scale range) are added, as this is useful for replaying generalisation operations progressively, while traversing the tGAP structure from top to bottom (in a strict sense, these extra pointers are not necessary, but this takes away the necessity to perform a translation of the face pointers using the tGAP face tree hierarchy to the high end of the scale range, before sending the edge)."

§ 3.3 Improved cartographic quality in constraint tGAP

The tGAP (topological Generalised Area Partitioning) structure was designed to store the results of the gradual generalisation process as described in previous sections. Earlier research had been focused more on proofing the concept, designing the solution, and shifting the technical limitation than improving cartographic quality of the tGAP. The generated content had been steered by an *importance* function, for selecting the least important object and then merge with the most compatible neighbour based on a *compatibility matrix*. From the cartographic point of view this solution was not really optimal.

The initial attempts for improving cartographic quality were carried out by Haunert et al. (2009); Dilo et al. (2009). Figure 3.6 presents an approach, named *constrained tGAP*. Instead of generating new maps from one source (Figure 3.6a) or a sequence of simpler and simpler map as in classical tGAP (3.6b), the process is based on a reference, smaller scale, well generalized map, see 3.6c. With the additional smaller scale input data (the constraints), the iterative algorithm can be controlled to obtain higher quality (intermediate) maps.



(a)UML diagram

```
CREATE TABLE tgap_faces (
face_id integer,
imp_low numeric,
imp_high numeric,
imp_own numeric,
feature_class_id integer,
area numeric,
bbox box2d);
```

(b)Face table

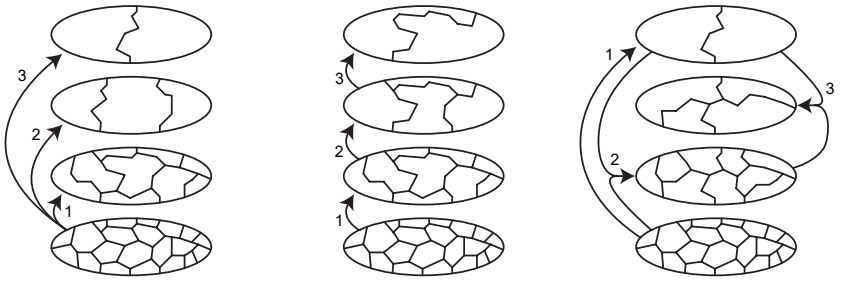
```
CREATE TABLE tgap_face_hierarchy (
face_id integer,
parent_face_id integer,
imp_low numeric,
imp_high numeric);
```

(c)Face hierarchy table

```
CREATE TABLE tgap_edges (
edge_id integer,
imp_low numeric,
imp_high numeric,
start_node_id integer,
end_node_id integer,
left_face_lowest_imp integer,
right_face_lowest_imp integer,
left_face_highest_imp integer,
right_face_highest_imp integer,
geometry geometry);
```

(d)Edge table

FIGURE 3.5 UML diagram and the database tables for tGAP, source: (Meijers, 2011, p. 159).



(a) Generalisation from a single source dataset (repeated optimization).

(b) Successive generalisation (classic tGAP approach).

(c) Intermediate representations fitting in the target representation (constraint tGAP approach)

FIGURE 3.6 Approaches to create a sequence of LoDs from (Haunert et al., 2009).

Generalisation for the tGAP data structure is performed on an area partition; thus the large-scale dataset is required to be an area partition. Area objects of the smaller scale data set act as region constraints in the generalisation process, *i. e.* they restrict the aggregation of large-scale objects only inside the region constraints. The method proposed here consists of two stages: the first stage matches objects of the large-scale dataset to objects of the smaller scale dataset, which act as region constraints in the next stage; the second stage compiles additional information needed for the constrained tGAP and performs the generalisation.

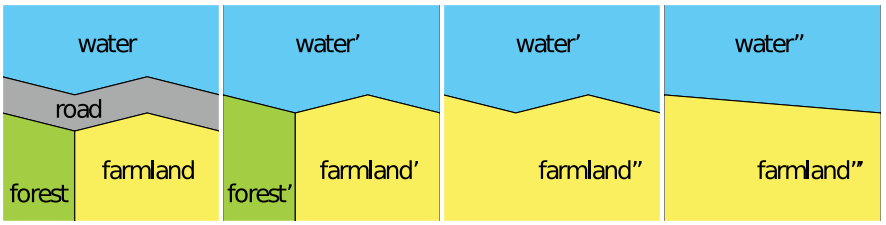
The result of generalization is highly dependent of the 'target' generalised dataset (smaller scale input) and it can be obtained via two routes: 1) from an external, independent source or 2) via a different generalisation algorithm (same source). The first option was explored by Dilo et al. (2009). The second option was investigated in (Haunert et al., 2009). Dilo et al. (2009) implemented constrained tGAP with real topographic datasets from the Netherlands for the large-scale (1:1,000) and independent medium-scale (1:10,000) data. This solution may suffer from geometrical and time inconsistencies, *e. g.* the geometry of reference scale is different or not up-to-date. This could make the preprocessing step even more complex.

On the other hand, Haunert et al. (2009) used German Land Cover data in 1:50,000 scale (ATKIS DLM50) as input. They eliminated the problem of inconsistency by deriving small scale data by the optimized (and computationally expensive) algorithm from the same base data.

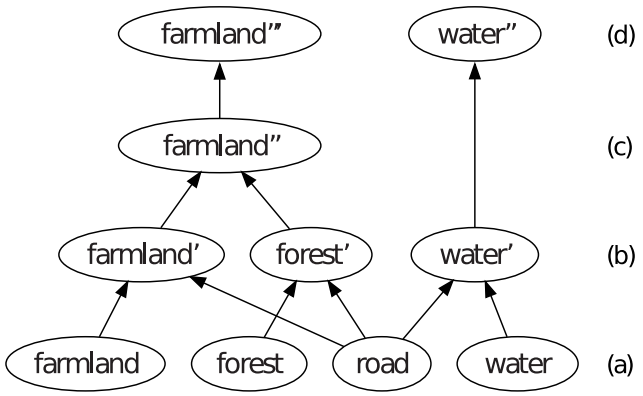
The most significant limitation of using constrained tGAP is in the fact of using additional knowledge derived from other solution. In theory, it means that the solution cannot run independently. Moreover, the results are very dependent on the 'target' small scale approach obtained from complex and expensive preprocessing step.

§ 3.4 The tGAP structure represented by the 3D Space-Scale Cube

The tGAP structure has been presented as a vario-scale structure (van Oosterom, 2005). So far, the historical overview of development and current implementation have been



(a)Original map (b)Result of collapse (c)Result of merge (d)Result of simplify



(e)Corresponding tGAP structure

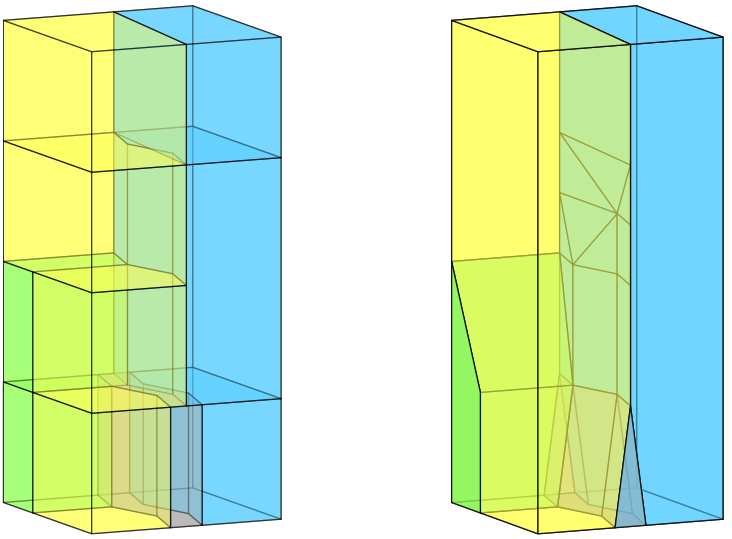
FIGURE 3.7 The 4 map fragments and corresponding tGAP structure (taken from (Meijers, 2011, p. 70)).

presented. Here, we will explain how more gradual (morphing-like transition) between the stages of the structure can be ensured.

The tGAP structure can be seen as result of the generalisation process and can be used efficiently to select a representation at any required level of detail (scale). Figure 3.7 shows four map fragments and the tGAP structure in which the following generalisation operations have been applied:

I. Collapse road object from area to line (split area of road and assign parts to the neighbours);
II. Remove forest area and merge free space into neighbour farmland;
III. Simplify boundary between farmland and water area.

The tGAP structure is a Directed Acyclic Graph structure (DAG) and not a tree structure, as the split causes the road object to have several parents; see Figure 3.7e. In current tGAP implementation the simplify operation on the relevant boundaries is combined with the remove or collapse/split operators and is not a separate step. However, for the purpose of this chapter, it is clearer to illustrate these operators separately. For the tGAP structure, the scale has been depicted as a third dimension – the integrated Space-Scale Cube (SSC) representation (Vermeij et al., 2003; Meijers and van Oosterom, 2011). We termed this representation the space-scale cube in analogy with the



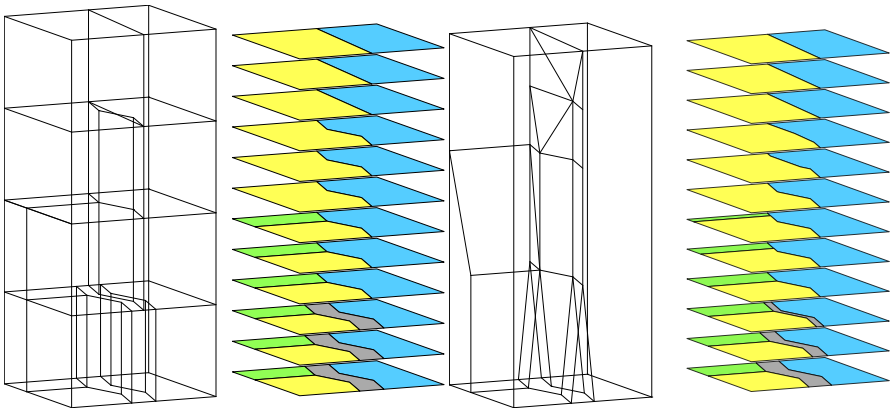
(a)SSC for the classic tGAP structure (b)SSC for the smooth tGAP structure

FIGURE 3.8 The space-scale cube (SSC) representation in 3D, source (Meijers, 2011).

space-time cube as first introduced by Hägerstrand (1970). Figure 3.8a shows this 3D representation for the example scene of Figure 3.7. In the SSC the vario-scale 2D area objects are represented by 3D volumes (prisms), the vario-scale 1D line objects are represented by 2D vertical faces (for example the collapsed road), and the vario-scale 0D point object would be represented by a 1D vertical line. Note that in the case of the road area collapsed to a line, the vario-scale representation consists of a compound geometry with a 3D volume-part and 2D face-part attached.

Though many small steps (from most detailed to most coarse representation – in the *classic tGAP*, $n - 1$ steps exist, if the base map contains n objects), this could still be considered as many discrete generalisation actions approaching vario-scale, but not truly smooth vario-scale. Split and merge operations cause a sudden local 'shock': a small scale change results in a not so small geometry change where, for example, complete objects disappear; see Figure 3.9a. In the space-scale cube this is represented by a horizontal face; a sudden end or start of the corresponding object. Furthermore, polygon boundaries define faces that are all vertical in the cube, *i.e.* the geometry does not change at all within the corresponding scale range, which result in prisms constituting a full partition of the space-scale cube. Replacing the horizontal and vertical faces with tilted faces as depicted in Figure 3.8b, results in *smooth tGAP* providing gradual transitions; see Figure 3.9b.

Then, a map can be obtained from this volumetric partition by horizontal slice. The continuously changing map one can imagine as gradually moving the slicing plane (and scaling) from the top of the cube downwards, there will be any object suddenly appearing or vanishing. All changes result in a smoothly changing 2D map: a small change in the map scale means a small change in the geometry of the resulting map.



(a) Wireframe and slices of classic SSC (with shocks).

(b) Wireframe and slices of smooth SSC.

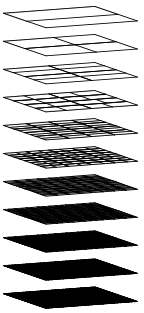
FIGURE 3.9 The map slices of the classic tGAP structure: (b) step 1 (collapse), (c) step 2 (merge) and (d) step 3 (simplify). Note that nothing changes until a tGAP event has happened. Figures originate from (Meijers, 2011).

§ 3.5 Mixed-scale representations

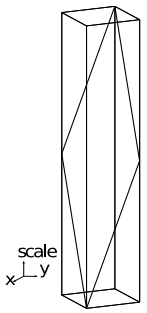
So far discussion has been restricted to the creation of 2D maps by slicing the SSC horizontally, however nothing prevents us from taking non-horizontal slices. Figure 3.10 illustrates a mixed-scale map derived as a non-horizontal slice from the SSC. What does such a non-horizontal slice mean? More detail at the side where the slice is close to the bottom of the cube, less detail where the slice is closer to the top. Consider 3D visualizations, where close to the eye of the observer lots of detail is needed, while further away not so much detail. Such a slice leads to a mixed-scale map, as the map contains more generalised (small scale) features far away and less generalised (large scale) features close to the observer.

The mixed-scale representation brings new aspects to consider:

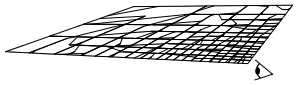
- *Non-planar slices* – 2D map can also be obtained by slicing surfaces that are non-planar; *e.g.* a bell-shaped surface that could be used to create a meaningful 'fish-eye' type of visualizations, see Figure 3.11.
- *Slices resulting in multiple parts* – It might be true that a single area object in the original data set might result in multiple parts in the slice (but no overlaps or gaps will occur in the slice). This increases the number of objects in the map and it results in a degradation of the quality of the map.
- *Limits of slicing surfaces* – What are other useful slicing surface shapes? Are there any limits? One can think of a bell-shaped, a sine curve surface or a 3D mesh object based surface. On the other hand, a folded surface seems to be nonsensical as it could lead to two representations of the same object in one map/visualization.



(a) A set of smooth slices derived from the SSC.

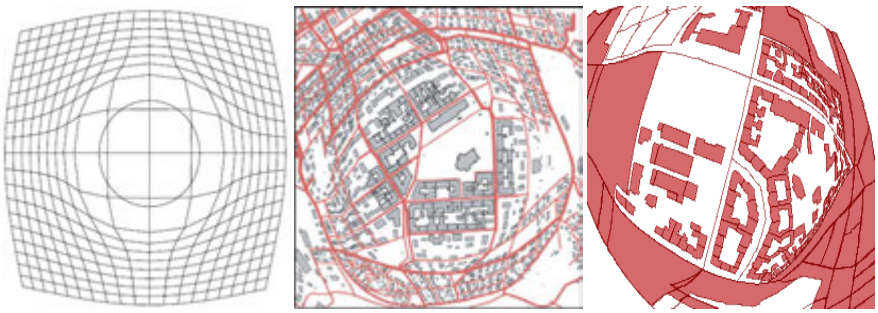


(b) How the non-horizontal slice of (c) is taken.



(c) Corresponding mixed-scale map (non-horizontal slice): top of map shows more generalised features than bottom.

FIGURE 3.10 Checker board data as input: each rectangular feature is smoothly merged to a neighbour. Subfigures show: (a) a stack of horizontal slices, (b) taking a non-horizontal slice leads to a 'mixed- scale' map and (c) one mixed scale slice (non-horizontal plane). All figures are taken from (Meijers, 2011).



(a) (b) (c)

FIGURE 3.11 A 'mixed-scale' map. Harrie et al. (2002) term this type of map a 'vario-scale' map, while we term this a 'mixed-scale' map. Furthermore, it is clear that there is a need for extra generalisation closer to the borders of the map, which is not applied in (b), but is applied in (c). With our solution, this generalisation would be automatically applied by taking the corresponding slice (bell-shaped, curved surface) from the SSC. Illustrations (a) and (b) taken from Harrie et al. (2002) and (c) from Hampe et al. (2004).

§ 3.6 State of the art

The text above gave overview about past developments and the state-of-the-art of vario-scale concept when PhD project started. There were also some open questions which indicate what were the main limitation and desires at that time, from (van Oosterom, 2005; van Oosterom and Meijers, 2012; Šuba, 2012). Below a list of open issues prior to this research is included:

I. It will be necessary to verify theoretical concepts about SSC more practically. Further, to test whether it is necessary to create and store SSC as a 3D structure or is it possible to work with only the 2D representation? How to realize conversion from tGAP to smooth tGAP?

II. Explore new possibilities of creating maps by slicing. It should be possible to create a map with a slice which is not even horizontal. It would lead to a 'mixed-scale' map. How can this be done efficiently? How will users perceive this?

III. The current structure only explicitly supports area features. Line and point features are not yet included explicitly in the storage structure. However these type of objects are produced during the creation of the tGAP structure. The collapse process is a good example. When the long feature is collapsed to a skeleton it changes dimension, *e. g.* from dimension 2 to dimension 1, from an area feature to a line. It is convenient to store information about the collapsed feature.

IV. Labels are an important part of maps, but are not included in the structure. Objects in the map need a label and every label in the map takes space. It is a classical cartographic problem: each object that has to be labelled allows a number of positions where the corresponding label can be placed. However, each of these label candidates could intersect with other label candidates and other objects as well. It is possible to find the right solution for just one predefined selected scale, but how can we find the right solution for the variable-scale?

V. Larger real world data sets should be tested (to further assess the potential of vario-scale maps). How should we deal with millions of records not fitting into the memory? One solution could be split in smaller parts. After that every part has to be computed separately (and potentially processed in parallel) and in the end these have to be merged together. How can the whole process be made manageable?

VI. One merge operation at a time is supported in the structure. More generalization actions could run concurrently, *e. g.* a parallel merging process could be an improvement with objects involved at different locations.

VII. Make the structure more dynamic. The current tGAP structure is a static one. It cannot handle changes of source data over time. If a small change of data takes place in the structure, the whole structure has to be recomputed.

VIII. Better decisions, because the selection of an object is based on area and class of object only. Moreover the thematic semantic aspect should be considered because there is no approach that would take into account the feature type of the objects, *e. g.* linear or areal.

TABLE 3.1 The list of open questions from previous publications (August 2012)

number	Proposed concept	Implemented, tested during PhD
I	SSC - practical test, convertion	yes, Chapter 6
II	Slicing, 'mixed-scale' map	yes, Chapter 6
III	Other map features	yes, Chapter 4
IV	Labels	no
V	Bigger dataset	yes, Chapter 5
VI	More operations simultaneously	yes, Chapter 4
VII	Dynamic structure	no
VIII	Better decisions	yes, Chapter 4
IX	Better techniques for gen. map content	yes, Chapter 4
X	Progressive transmission	no, §3.7
XI	User testing	yes, Chapter 6

IX. Improve the current vario-scale techniques for better vario-scale map content. Can we come up with a better method to steer the generalization process that is used to obtain data for the tGAP data structures?

X. Explore progressive transmission (and implement a working prototype in a GIS viewer, such as QGIS). A streaming solution, in which previously sent ordered geometry is reused, will only be possible with an approach where redundancy is addressed from the start. Data streaming becomes more important with a very large dataset. Having a coarse overview first, adding gradually more detail, makes it possible to stop when sufficient detail has arrived.

XI. Measure the impact of vario-scale on users. Is the vario-scale approach perceived better as multi-scale solution or faster, more intuitive or more effective for user navigation. User testing is a tool for such a validation. However there is no prototype to realize such testing at this moment.

These open questions give us an idea what were the most urgent issues in the project. However, the number of problems faced was quite large and the research time was limited. Therefore we considered; 1) the original research proposal, 2) discussion with the STW Users Committee, 3) a logic order, and 4) urgency in the vario-scale research. Those aspects have been used for defining the main research parts in the next chapters. Table 3.1 summarize the list above. It also indicates what was selected to be researched in this thesis; If so, the related chapter is mentioned.

§ 3.7 Progressive data transfer

Table 3.1 indicates that topic *progressive data transfer* was investigated during the time of this PhD project. Even though the topic is not within main focus of this work, it is important to mention it because shows additional aspect of vario-scale approach and completes the picture of the project.

Huang et al. (2016) have shown a solution for the progressive transfer of geographic vector data using vario-scale data structure in web-services setting. Note, that no 3D SSC and slicing as described in Section 3.4 was implemented, it is based on 'normal tGAP'. It focuses on data streaming through possibly limited bandwidth. First, the

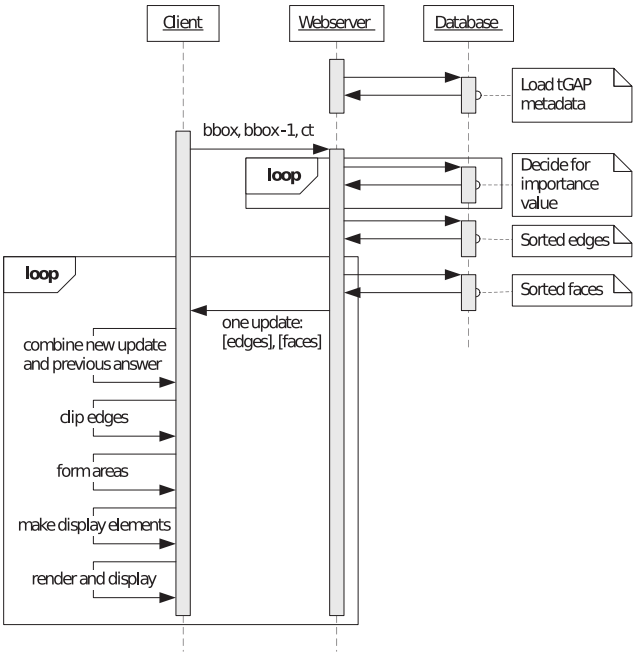


FIGURE 3.12 Client-server communication architecture for progressive data transfer (Figure 9 of (Huang et al., 2016)).

vario-scale data structure is prepared on the server side which encodes the results of the sequence of generalization steps; then, client-server communication architecture with client-side visualizations and smooth zoom interactions is efficiently realised using data from a vario-scale server.

In situation when the user performs an zooming operation, see Figure 3.12, the request parameters are; 1) a pair of bounding boxes (capturing the original and desired map state) and 2) optimal counts[1]. The response consists of a stream of chunks. Each chunk contains sufficient information (*i. e.* edges, faces and importance value) to bring the map at the client side to a new consistent state. Thus the map is gradually refined with content after an updated edges and face have been processed –- the user might still be zooming. Therefore, the response is dealt with chunk by chunk. For every incoming chunk of the response:

I. The client selects which edges and faces have to remain from the last response received.
II. The winged edge structure is updated accordingly (adding new edges/faces, removing unneeded faces/edges).

1 Optimal count defines the user preference of how much data is retrieved and visualised. By changing it, the user can modify the map appearance from a dense map to a map with few objects.

III. Ring and display objects are formed for updated topology primitives (faces) and the relevant styling is added.

IV. Display objects from the previous step are updated accordingly (those SVG polygons that are no longer needed are deleted and new ones that were formed are added).

This leads to faster visual feedback for a user. The user can see the map improve while downloading additional refinements (with a partial answer the map can already be constructed).

The idea of vario-scale as an optimal tool for progressive transfer of geographical data via the Internet was one of the original hypotheses, but it has been never proven until now. The paper by (Huang et al., 2016) demonstrated the potential of the vario-scale concept, because solutions for efficiently transferring vector data especially for limited bandwidth are urgently needed in desktop and mobile applications at high performance levels. To conclude, this makes the vario-scale approach relevant research in the field of automated map generalization because it combines 1) minimal duplication in data storage, 2) explicit links (because of the hierarchy structure) between the map objects, and 3) support for progressive data transfer.

§ 3.8 Final remarks

This chapter presented vario-scale development in the past and state of the art when PhD project started. It explained vario-scale approach in more detail using a specific data structure, which stores the results of map generalization actions; features are generalized in small steps, progressively leading to simpler and simpler maps. The main advantages of the concept compared to the multi-scale approaches can be summarized in the following aspects:

I. Redundant storage is eliminated as much as possible by avoiding duplication of features (as in multi-scale) and storing the shared boundaries between neighbouring areas instead of the explicit polygons themselves, *i. e.* using a topological data structure composed of nodes, edges and faces.

II. More intermediate representations can be constructed, because the automated generalization process records for every topological primitive (node, edge or face) the valid range of map scales for which this element should be shown.

III. The lineage of the generalization steps is stored explicitly and implies the links between generalized and original objects. Obtaining these links is regarded as a very difficult task for multi-scale databases. The main advantages of this are (1) making progressive updates/transmission possible and (2) future updates of the features are more easy to implement without re-generating the dataset.

IV. The Space Scale Cube (SSC) conceptual model makes possible really smooth transitions throughout the scales. It changes representation from 2D map features to 3D polyhedra. Then to derive a map from an SSC, the model is sliced by a plane at the desired level of detail. Hereby, exact LODs can be chosen and smooth transitions between abstraction levels are ensured.

Our approach can, in some aspects, already provide an advanced solution comparable to results of a well-known multi-scale solution. However, the vario-scale concept is still ongoing research which requires further development and testing. Therefore, Section 3.6 identified the most urgent needs of the approach and some of them will be addressed later.

For the remaining chapters it is good to keep in mind that vario-scale produce a sequence of successively more generalized maps, so that these maps go well together, similar to (Chimani et al., 2014); instead of considering each level of generalization independent. We aim at providing more degrees of freedom in data use. In our perspective, the sequence of generalization steps is more valuable than just the final map of specifically-identified scale. Our ambition is to produce data without explicit lower and/or upper bound. This makes the method very generic. On the contrary, it is difficult to compare with current maps because vario-scale is different by design.