TECHNISCHE UNIVERSITÄT DRESDEN

Derivation of continuous zoomable road network maps through utilization of Space-Scale-Cube

by

Meysam Aliakbarian

A thesis submitted in partial fulfillment for the
Master of Science in **Cartography**

in the
Faculty of Environmental Sciences
Department of Geosciences
Institute for Cartography

October 2013

Declaration of Authorship

I, Meysam Aliakbarian, declare that this thesis titled, 'Derivation of continuous zoomable road network maps through utilization of Space-Scale-Cube' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly during my thesis period at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

"You will learn by reading, but you will understand with love."

Jalal ad-Din Muhammad Balkhi,
AKA **Molana**,
AKA **Mowlavi**,
AKA **Rumi**.
Persian/Iranian poet and mystic, (1207 - 1273 A.D.)

TECHNISCHE UNIVERSITÄT DRESDEN

Abstract

Faculty of Environmental Sciences
Department of Geosciences
Institute for Cartography

Master of Science

by Meysam Aliakbarian

The process of performing cartographic generalization in an automatic way applied on geographic information is of highly interest in the field of cartography, both in academia and industry. Many research efforts have been done to implement different automatic generalization approaches. Being able to answer the research question on automatic generalization, another interesting question opens up: "Is it possible to retrieve and visualize geographic information in any arbitrary scale?" This is the question in the field of vario-scale geoinformation. Potential research works should answer this question with solutions which provide valid and efficient representation of geoinformation in any on-demand scale. More brilliant solutions will also provide smooth transitions between these on-demand arbitrary scales. Space-Scale-Cube (Meijers and Van Oosterom 2011) is a reactive tree (Van Oosterom 1991) data structure which shows positive potential for achieving smooth automatic vario-scale generalization of area features. The topic of this research work is investigation of adaptation of this approach on an interesting class of geographic information: road networks datasets. Firstly theoretical background will be introduced and discussed and afterwards, implementing the adaptation would be described. This research work includes development of a hierarchical data structure based on road network datasets and the potential use of this data structure in vario-scale geoinformation retrieval and visualization.

Acknowledgements

When I look back to the time when I started my Master studies, I see a lot of people who affected this path from that point till now. There would be a long list of many people to name, but hereby I can just name a few of them.

I would like to thank our Masters program coordinator Dipl.-Ing. Stefan Peters of TUM for his brotherly non-stop help before we start our Master studies until the last day. Besides Stefan there are professors and staff members of three universities who have been involved in our Masters program, to whom I am very thankful. Specially I would like to thank Professor Georg Gartner of TU Vienna for his positive and energetic attitude which is always full of impressions.

During my Master thesis period I had the chance to work with both of my supervisors Professor Dirk Burghardt and Dipl.-Ing. Karsten Pippig of TU Dresden. This chance provided me a lot of experience and impression. I am really thankful to both of my supervisors for their guidance, patience and support. I also thank Dr. Martijn Meijers of TU Delft for his openness and help in the beginning of this thesis.

But life is not only around studies and research. There are always people who give you love and motivation to work hard and achieve whatever you want. I am very grateful to my cousins Soraya Ali-Akbarian and Stephan Bock, without whom I was not able to be where I am now. Also I would like to thank my old friends Hamid Amrollahi and Mehdi Bahrami who gave me their kind remote support.

October 2013

Meysam Aliakbarian

Contents

List of Figures

Abbreviations

API	**A**pplication **P**rogramming **I**nterface
BLG	**B**inary **L**ine **G**eneralization
DCM	**D**igital **C**artographic **M**odel
DLM	**D**igital **L**andscape **M**odel
GAP	**G**eneralized **A**rea **P**artitioning
GDAL	**G**eospatial **D**ata **A**bstraction **L**ibrary
GIS	**G**eographic **I**nformation **S**ystem
GML	**G**eography **M**arkup **L**anguage
GUI	**G**raphical **U**ser **I**nterface
ICA	**I**nternational **C**artographic **A**ssociation
JSON	**J**ava**S**cript **O**bjec **N**otation
KML	**K**eyhole **M**arkup **L**anguage
LoD	**L**evel **o**f **D**etail
MBR	**M**inimum **B**ounding **R**ectangle
MRDB	**M**ultiple **R**epresentation **D**ata**B**ase
NMA	**N**ational **M**apping **A**gency
RDBMS	**R**elational **D**ata**B**ase **M**anagement **S**ystem
SQL	**S**tructured **Q**uery **L**anguage
SRID	**S**patial **R**eference system **ID**entifier
SSC	**S**pace **S**cale **C**ube
SVG	**S**calable **V**ector **G**raphics
tGAP	**t**opological **G**eneralized **A**rea **P**artitioning
WKB	**W**ell **K**nown **B**inary
WKT	**W**ell **K**nown **T**ext

To my family:
Afsaneh, Hossein, Maryam and Misagh.

Chapter 1

Introduction

With rapid development of hardware, software, services and network connection speed world-wide, there is a high rate of production, transmission and consumption of information. As human being life is based on many constraints like space, time and energy, making more efficient decisions are of higher interest in order to consume less resources. Since geography plays an important role in understanding people surrounding, geographic information is an important player in the information era. In Geographic Information Systems (GIS), geographic data is being stored, manipulated, retrieved and published. As mentioned by many authors (O'Sullivan and Unwin 2003; McMaster and Shea, 1992), during the life-cycle of GISs almost any operation based on geographic information has some dependency on scale of data.

By definition the process of deriving smaller scale geographic/cartographic representation from larger scale information is called cartographic generalization (Nickerson 1988). In the past the process of cartographic generalization have been done by professional cartographers doing the abstraction and applying different generalization operators on the data. The decision criteria, composition of operators and processes; and result evaluation have been achieved through practice and experience. Performing such time and resource consuming process should be done using computer systems. Scientific background and results of this field of research will be discussed later in chapter 2. But a brief background would be discussed here to introduce shortly and clarify the scope of this document.

Modern generalization methods imply the idea of deriving multiple Levels of Detail (LoD) and then storing the result in different databases (Digital Landscape Models, DLMs). By having appropriate abstracted data there would be possibility to visualize data in an appropriate visual representations (Digital Cartographic Models, DCMs) as in (Grüreich 1985; Brassel and Weibel 1988). Information of different models are stored

while cross-linked features who point to the same entity in different models. These methods tend to be a solution for the problem but according to Van Oosterom (2009) there is a main disadvantage which is data redundancy and having data and features in several geographic database systems. Geographic features storage and relations would be implicit and redundant while building and maintaining these relations explicitly seems impossible in a long run process. Having DLM-DCM or Multiple Representation Database (MRDB) systems, development of some methodologies who store geoinformation once and without any data redundancy would be still of high interest. Reactive data structures introduced by Van Oosterom (1991) are set of possible answers to this demand. There have been several implementations of reactive data structures and the present research work is the first member of this group of data structures which applies on road network data structures.

1.1 Problem Definition

Considering different efforts of finding means of automatic vario-scale generalization, Space-Scale-Cube or abbreviated as SSC is a promising effort to result in a fast, efficient and *vario-scale* potential data structure which simplifies storage and querying geographic information. Studying the literature concerning the idea, one would come to this conclusion that every effort in this field has been done in order to implement a data structure appropriate to query area information defined by the space surrounded by a closed polygon. Area information is an appropriate model to visualize areas, cadastre, county regions, provinces, federal states, countries and etc.
On the other hand there is also need to model connections between places. In real world this is the background of having streets and roads. Such objects are modeled using networks in a macro level view. In GIS every section of a road network is modeled and described using line features (polylines or linestrings) who are also connected to semantic information of that member of network. Generalization of networks and specially road network datasets is of high interest in several fields (e.g. traffic visualization, route planning, navigation systems and etc.). Considering all of these concepts together, several new questions open up in this research field.

1.1.1 Research Questions

Considering all of the above information and assumptions:

1. Is it possible to design and implement algorithms and sufficient data structures to perform vario-scale generalization on data with road network nature?
2. Is it possible to perform transition of set of road network information to a reactive tree data structure?
3. Is is possible to build a hierarchical parent-child relationship appropriate for tree data structure?
4. Is it possible to implement a smooth transition between different LoDs?

1.1.2 Objectives

The research work should satisfy the following objectives:

1. Develop theory for adapting tGAP (topological Generalized Area Partitioning) and SSC (Space Scale Cube) data structures for road network datasets.
2. Implement, test and evaluate the suggested theory of road network reactive data sources.

1.2 Proposed Solution

Having clear research questions, the proposed solution is going to be discussed. In general this research work is divided into two parts, *theoretical solution* and *practical solution*; both of them will be discussed in details in future chapters (please refer to the following section for further details). For the theoretical part, reading the background literature of continuous generalization, tGAP, SSC and network generalization, a suggested algorithm of forming a reactive tree data structure will be discussed. The algorithm builds a hierarchical structure from the network with consideration of the connectivity nature of the network. It uses the topological formulation of connectivity in networks using graph theory. Having the data structure in theory, this enables the implementation in a programming environment. The implementation queries the hierarchical tree data structure and performs traversing on the tree in order to get appropriate data. There would be also means of visualization of the fetched data to potential user of the implemented system.

1.3 Structure of the Thesis

This research work is presented in 5 chapters:

- **Chapter 1**: The current chapter; which gives introduction about the research work, background and the structure of this document.
- **Chapter 2**: Consists of discussion about cartographic generalization in general and some more focus on two topics: continuous generalization and network generalization.
- **Chapter 3**: The developed theory for road network SSC is discussed to clarify the attributes and behavior of this data structure.
- **Chapter 4**: Practical implementation of the road network SSC is described.
- **Chapter 5**: The outcomes of the research are being under discussion and some suggestions are given about future steps in this field of research.

1.4 Notes on Terminology

As the name suggests, topological Generalized Area Partitioning (tGAP) is a data structure for applying generalization algorithm on area features. The author is aware that road networks are set of line features and are normally modeled by polylines in GISs. Using the same terminology is because of loyalty to the terminology of the original concept and also helps the potential readers to understand and adapt the new concept easier.
It should be also mentioned that the idea of Space Scale Cube (SSC) is also better understood as Space Scale Pyramid while imagining the structure being more generalized (and thus in smaller scale) when moving toward the positive values of the 3rd dimension (scale factor) of the corresponding 3D geometry. Again there is no change in the terminology for loyalty and better knowledge transfer.
There should be also clarification of other several terms which are being mentioned in this research work and may result in confusion of readers. The terms *point*, *vertex* and *node* are commonly used. If not explicitly mentioned otherwise: *point* refers to a zero dimensional geographic feature who has (x, y) coordinates; *vertex* refers to a knot in a graph and *node* is an interchangeable word which may relate to any of the two mentioned terms. There is also the term *tree-node* which is a cell in a tree data structure. Regarding other commonly terms *segment*, *polyline*, *linestring* and *edge*: *segment* is a

line who connects two or more points to each other; *polyline* and *linestring* refer to geometric and/or graphic representation of one or more line(s) and *edge* may refer to either *segment* or a connection who connects two vertices in a graph.

Chapter 2

Cartographic Generalization

According to Weibel (1997) generalization is an important key process in the field of cartography, many research attempts have been done to automate this process through cartographic activities. Performing such key element using computer systems is a highly active field of research in modern cartography.
This chapter will introduce and discuss several important and effective; proposed and developed approaches in the past. The discussion starts with definition of generalization and classification of generalization algorithms. Afterwards a review of well-known generalization operators takes place. Then a review of research works related to continuous generalization (or vario-scale geoinformation) is provided. The discussion continues with background of generalization of networks. At the end of this chapter the area of interest of this research work is being mentioned.

2.1 Cartographic Generalization: Definitions and Classifications

As suggested by many authors including Longly et al. (2005); O'Sullivan and Unwin (2003), geographic information is very dependent on scale. This statement has the meaning that storage, process and representation of such information in medium of paper maps, modern computer systems (GIS) and modern mobile devices is also highly dependent on scale. An official definition from International Cartographic Association (ICA, 1973) introduces generalization as "the selection and simplified representation of detail appropriate to the scale and/or purpose of a map".
Generalization can be also formulated as according to McMaster and Shea (1992), by answering 3 basic philosophical questions. Then answers will define the generalization process:

- Why: Consideration of the objectives of process.
- When: Carto-metric evaluations for input and outputs of the process.
- How: Selection of the transformation of data from grater scale to smaller scale.

Also an early definition of generalization by Töpfer and Pillewizer (1966) suggests "Generalization covers process of selection and simplification and the reason is the physical constraints of the map (the size), which limits the amount of objects to be shown at a certain scale of interest.". The process of generalization used to be done by professional cartographers but now the interest is about doing it through automatic systems. There would be need to understand, formulate, model, repeat and evaluate generalization or as it has been mentioned by Brassel and Weibel (1988), it should be broken down into five steps: structure recognition, process recognition, process modelling, process execution and display. This sequence is result of separation of two concepts: *characteristics extraction* which determines the generalization means and *modification process formalization* to be performed on features.
To give a more clarified definition and to formulate this process several researchers have broke down generalization to other easier understandable definitions. Raisz (1962) simplifies the definition of cartography to *Combination*, *Omission* and *Simplification* of map entities. This definition is more to define generalization through operators behavior. Furthermore Robinson, Sale and Morrison (1978) classify generalization field to two main founding entities: *elements* and *controls*. Elements of generalization are: *Simplification*, *Classification*, *Symbolization* and *Induction*. Beside elements, there are also four control entities mentioned by these authors which are: *objective*, *scale*, *graphics limit* and *quality of data*.
According to Müller (1991), generalization can be seen as a process which takes use of one or more operations or as suggested by Grüreich (1985) it can be seen as a chain of processes leading to one or more models as data storages. Several formalization of frameworks have been done to help researchers (like Brassel and Weibel 1988; Steiniger and Weibel 2005). These efforts would help potential researchers to achieve more understanding and an overview of available algorithms and data models.
According to widely used reference among authors (Grüreich 1985; Müller, Lagrange and Weibel 1995; Meijers 2006; Galanda 2003; Cecconi 2003), generalization can be divided into three parts which convey both different chronological stages and also abstraction levels:

- Object generalization: Abstraction with a finite amount of observations and measurements from the objects in reality to a database/model correspondent. Mainly

is abstraction but also can be seen as selection and reduction of real object and leads to a data model (primary DLM).

- Model generalization: Applying spatial and semantic transformation with a controlling manner in order to result in a set of less features and resulting in reduction of memory and storage usage. The result is a dataset appropriate for faster transfer and more efficient processing which can be an input for the later steps. The model is named a secondary DLM.

- Cartographic generalization: Mainly concerned with improvement of the visual representation of data in form of digital or paper maps. The main role is played by symbolization and the result is a DCM. Visual legitimacy and satisfying cartographic and semantic restrictions is the main controls of this process.

This framework has been suggested by Grünreich (1985) and later has been adopted by Weibel and Dutton (1999) and is shown in figure 2.1. According to many authors

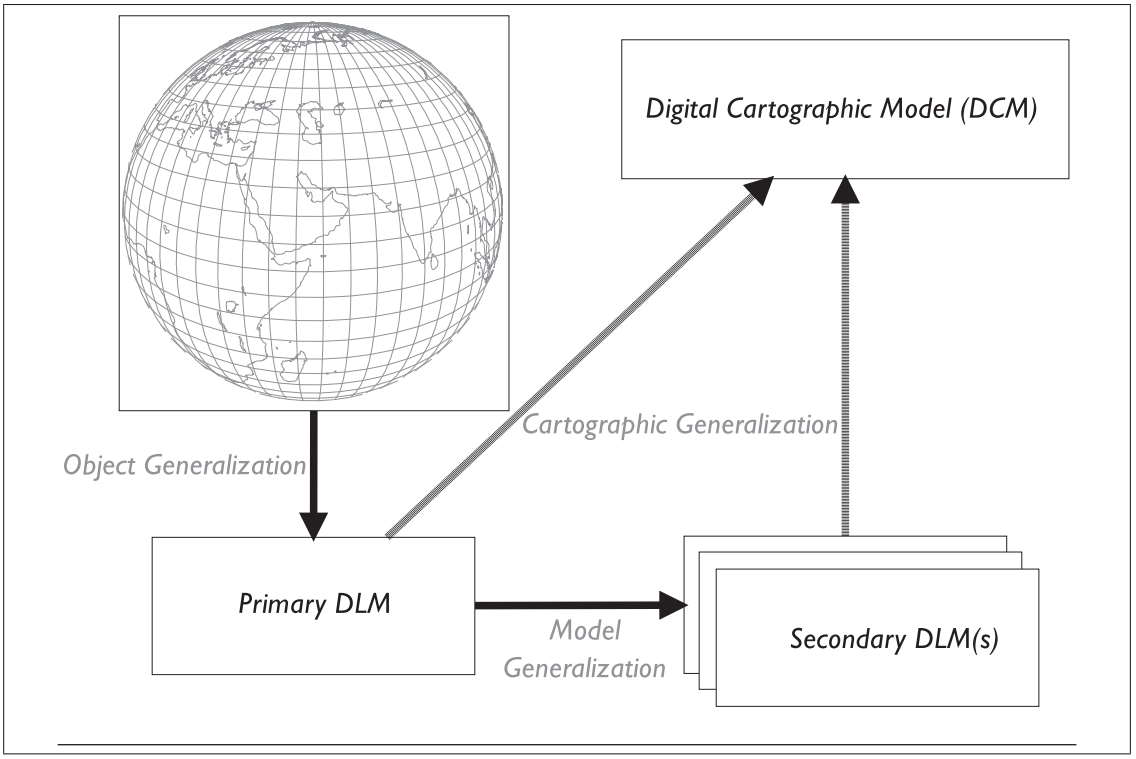

FIGURE 2.1: DLM-DCM generalization model.

(Beard 1991; Weibel and Dutton 1998; Ruas and Plazanet 1996; AGENT 1998; Peter and Weibel 1999; Burghardt, Schmid and Stoter 2007), there is another important defining aspect of generalization which should be considered in research: there are always several constraints which should be satisfied and in most cases there are more than one constraint. According to Peter and Weibel (1999), the main role of a constraint is to limit the number of possible solutions without binding it to a particular action. Beard (1991)

groups constraints to four categories: *graphic*, *structural*, *application* and *procedural* while according to Ruas and Plazanet (1996) the categorization of object constraints covers: *legibility*, *shape*, *spatial* and *semantic*. Another categorization of constraints is from project AGENT (1998) which divides constraints in five categories: *graphic*, *topologic*, *structural*, *gestalt* and *procedural*.

There are several shared aspects mentioned in these definitions:

- Generalization is understood to be a process which is done through different operations. Different operations and operators can work individually or in a chain or in a more complicated mixed process.
- The operation is mainly in order to abstract and reduce data itself and complexity of data.
- There are several controls or constraints who should be satisfied.
- Several characteristics of map features play role in the process itself or in control and/or evaluation of the results. The characteristics can be of geometry, coordinates, semantics, structure or topology type.

2.2 Generalization Operators

One important role-playing aspect of generalization is the operator. By modeling human behavior of generalization, several primary operators have been identified (according to McMaster and Shea 1992; Foerster, Stoter and Kobben 2007; Robinson, Sale and Morrison 1978):

- (Class) Selection: selection the specific instances of a specific feature type, which should appear in the target data model. The process is normally done based on a query parameter or an attribute of such instances. The opposite process is considered as Omission or Elimination.
- Simplification: The process of reducing the geometric complexity of features toward a simpler representation. This is normally done through selection of a subset of the geometry of the original feature.
- Amalgamation: Amalgamating a set of spatially adjacent/close geometries of the same class into a single geometry which leads to higher abstraction. There are also other names for the same behavior or in some cases with slightly different behavior in detail, namely: Aggregation, Merge and Fusion.

- Collapse: Is the process of reducing the geometrical dimension of the feature to a lower dimension.
- Displacement: Overcoming complication from proximity, overlap and coincidence by displacing features in order to improve distinguishability.
- Refinement: Refining the structure features to filter less critical features and represent more critical features to reduce the complication.
- Smoothing: Capturing more important trends of the lines while dismissing less important trends which would result in less sharp angularity of the geometry.
- Exaggeration: Applying geometric exaggeration on certain features in order to guarantee visibility of certain geometric characteristics of the feature. Normally reducing the scale will affect some visibility detailed criteria of features.
- Enhancement: Modifying certain geometric parts of graphic representation to produce a pleasing view or to emphasize an object in order to fit the requirements of the map.
- Classification: Grouping features who share an identical or similar attribute into categories.
- Symbolization: Assignment of symbols to class features in order to represent the similarity between the sets.

2.3 Efforts on Vario-Scale Visualization of Geoinformation

Considering demand for generalization of paper maps or the DLM-DCM and MRDB models of geographic information storage, a generalization process can be carried out to derive small scale data from large scale data. The source is of higher scale with detailed information and the product has smaller scale with more abstracted information. National Mapping Agencies (NMAs) have to deal with different production scales of interest for potential maps (paper print product). Even with modern cartographic mediums, scale of visualization is seen as a discrete quantity. As a proof, short check on online web mapping service interfaces and even their APIs would support this conclusion.

But there has been always interest concerning solutions which enable implementation of a system supporting geoinformation visualization in any arbitrary scale. This interest is among scientists, cartographers, NMAs, GIS software vendors and finally map users. In such systems the scale has no predefined values and the data query would ask for

geoinformation of any scale to get a visualization from the system. Several efforts will be discussed here but as several researchers have worked in this field, different terminology of methods consists in the literature but in general there are similar concepts behind them. Some different terminologies are as follows:

- *Scaleless database* as in Van Oosterom (1991).
- *On-the-fly generalization* as in (Van Oosterom 1995; Burghardt, Purves and Edwardes 2004; Cecconi and Galanda 2002).
- *Continuous generalization/zooming* as in (Van Kreveld 2001; Sester and Brenner 2005).
- *Vario-scale or variable scale geoinformation* as in (Meijers and van Oosterom 2011; Meijers 2006; Van Oosterom 2005).

According to Van Kreveld (2001), smoothing between map scales can be thought of as cartographic animation in which the change is based on scale rather than time; which support natural and aesthetical visualization. The suggested effort in his research is about applying smooth changes between different levels of generalization. Considering a linear relation between scale and amount of generalization, several smoothing strategies for inter-LoD transitions have been suggested for some generalization operators (displacement, elimination, simplification). By these means, having different LoDs and through a query there would be the possibility to derive an appropriate visualization based on an LoD of interest. For implementing such system there would be also necessity of formalizing inter-LoD transitions for any generalization operator. Thus having different LoDs and smoothing strategy for specific generalization operator would make vario-scale maps reachable. Instead of performing transitions, a process of iterative additional/elimination of line segments through formulating processes for inter-LoDs has been discussed by Sester and Brenner (2005) with the aim of visualization on mobile devices with small screens.
In order to derive and store LoDs in a data storage, there have been several efforts by different researchers. The focus is to avoid data redundancy introduced by geographically overlapping data in different LoDs, mentioned by Van Oosterom (1991, 2009). The reactive-tree is introduced by Van Oosterom (1991) and tries to provide a data structure who has two main characteristics: spatial capability (indexing) and capability of storing multiple LoDs. This effort assigns importance values to area features in order to differentiate them. Such importance values are stored and later are used to fetch appropriate features who fit visualization query criteria. The same concept of tree formation and storage is used on later similar works by Van Oosterom (GAP 1995; tGAP

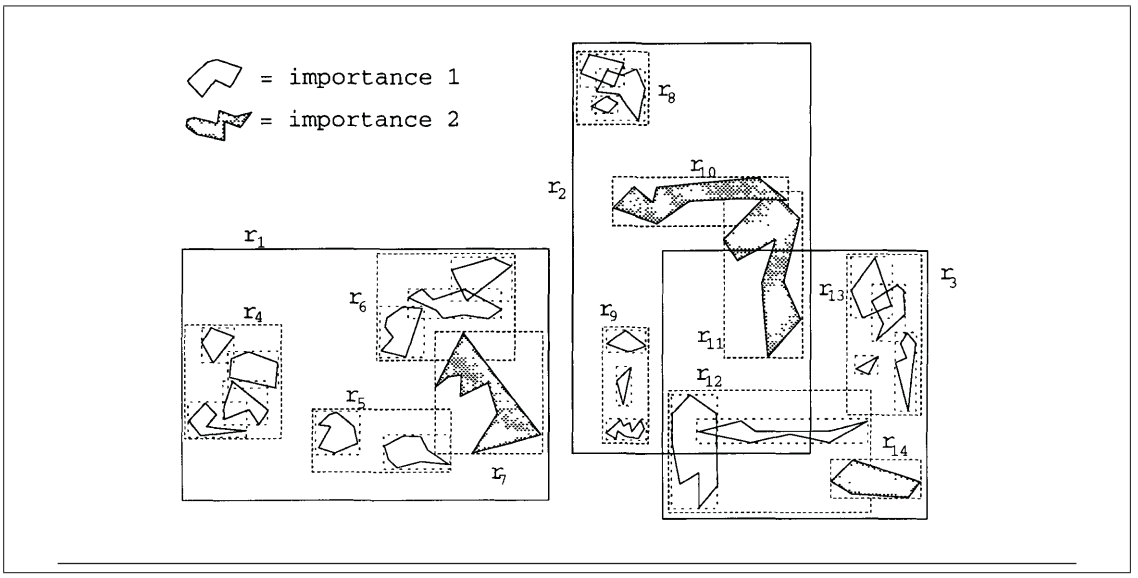

FIGURE 2.2: An example of a region which is imported to a Reactive-tree structure. Figure taken from Van Oosterom (1991).

2005). The entities in a reactive-tree structure are of two types: *objects* and *regions*. Objects are the geographic objects who are being mapped in the structure and regions are pseudo-objects which are R-Tree (Guttman 1984) regions. This means that every object is being formulated by its MBR (Minimum Bounding Rectangle) and regions are only a set of objects who only hold a union of their children MBRs. An example of a scene and its corresponding reactive-tree can be seen in figures 2.2 and 2.3. A

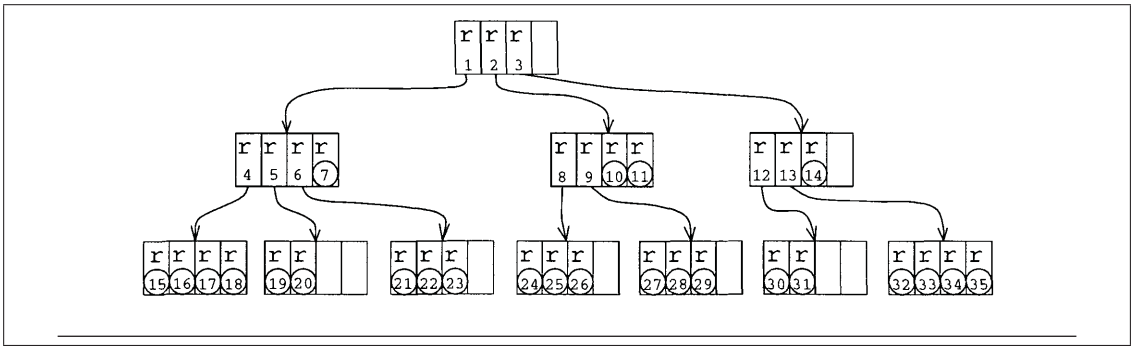

FIGURE 2.3: The same region is imported to the tree structure. Figure taken from Van Oosterom (1991).

development of reactive-tree is implemented in GAP-tree (Van Oosterom 1995). The concept of having a tree structure which is going to be traversed to get the features of the map remains the same. In GAP-tree structure the R-Tree indexing is replaced by coverage of the features. This means that there would be no more tree nodes without corresponding object in reality and every node of the tree is corresponding to an area feature. In GAP-tree data structure, formation of the tree is based on elimination of least important faces (2D area features). The orphaned characteristics of the eliminated feature and its liberated area is assigned to a neighbor with the most similarity. Again

the tree structure is being used to store the relation of features and later it is being used to retrieve appropriate data with the query parameter. Criteria of the data query is based on importance values which are assigned to features in the beginning. After weeding of the scene to result in selection of more important features, the importance values are hierarchically assigned to higher rank features of the tree. Figure 2.4 shows an example of this data structure. Further improvement of the GAP-tree structure is

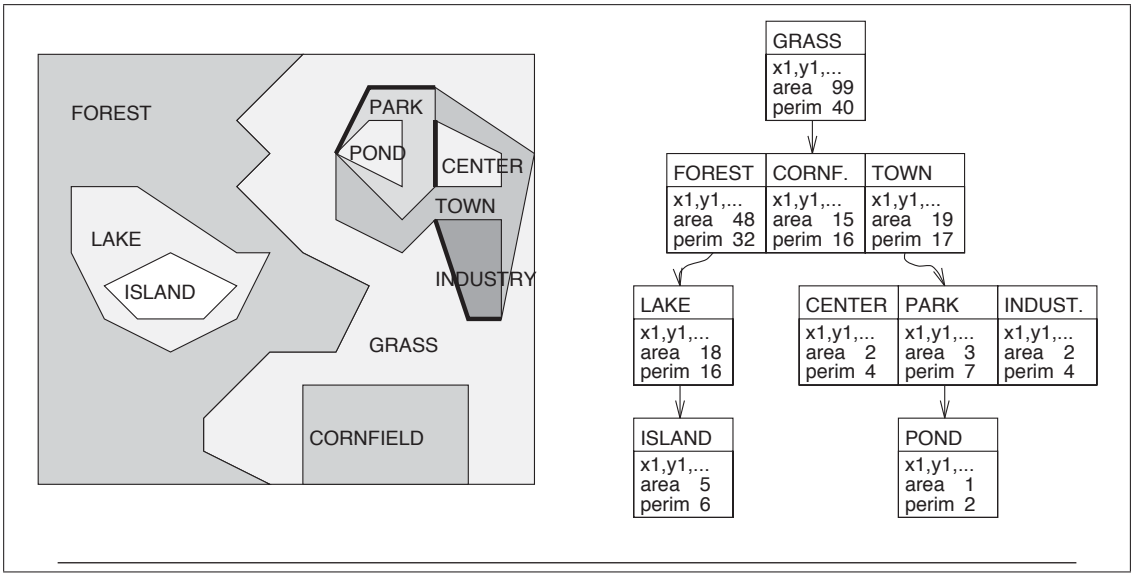

FIGURE 2.4: A scene of interest (left) and its GAP-tree correspondent data structure (right). Figure taken from Van Oosterom (1995).

followed in tGAP data structure (Van Oosterom 2005). tGAP is intended to resolve problems introduced by GAP-tree: redundancy issues. The latter data structure is a combination of trees: a GAP-tree which stores the relation of the faces (area features) and a multitude of GAP-trees storing edges (set of trees is called a forest). tGAP is a topologically-aware data structure holding edge-face topological relations. The face tree acts similar to the original GAP-tree and holds the relationships of area features along with their assigned and calculated importance values. The edge-forest is the storage of edges hierarchical relations. The same operations of elimination and assignment to parents take place with the edges. Studying hierarchical relations among area features in comparison with hierarchical relations among edge features, results in an important conclusion: there is no possibility of representing all edges with one edge feature. In the case of area features, at the end of the elimination process there would be a survived area which covers the whole area of the start set of data. That feature would be the root of the face-tree in tGAP. In case of edges, there are possibilities that two neighbor area features are being merged. The border of these area features simply disappears from the next step (without receiving any feature as parent). Having no unique root implicitly means that more than one tree is needed to store edges relations. The general process elimination and merge is shown in figure 2.5 and the edge forest formation is shown

in figure 2.6. As it was discussed with reactive-tree, GAP and tGAP structures, the

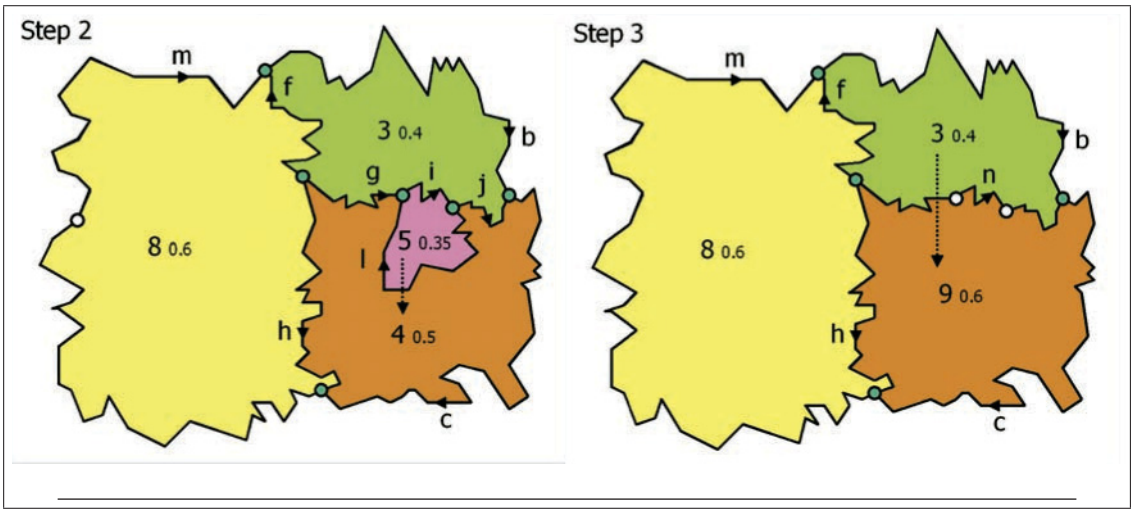

FIGURE 2.5: tGAP formation process. Two steps and the decisions for the next steps are shown. Pay attention to *i*, *j*, *g*, *n* and *l* edges. Figure taken from Van Oosterom (2005).

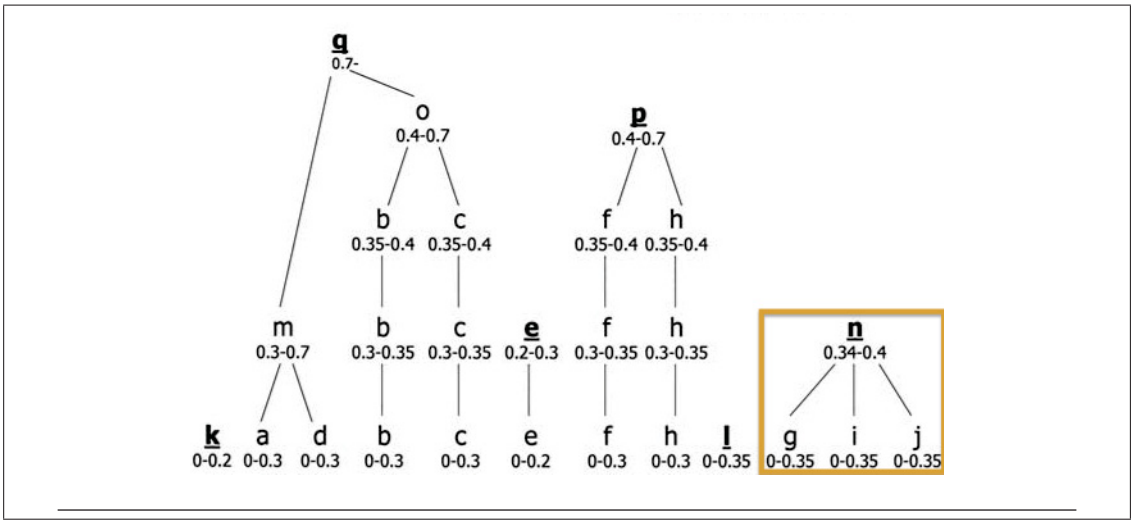

FIGURE 2.6: The corresponding edge forest is shown. The highlighted area shows the edge changes from last figure. Figure taken from Van Oosterom (2005).

role-playing factor for selection of features is an assigned or inherited importance value. It determines whether a feature would be in the result set of the visualization query or not. While combining the idea with a geoinformation visualization environment, this importance quantity can be inter-transformed to scale factor of the modeled geoinformation. As the goal is a vario-scale system, there would be possibility of continuous values of scale to query the data structure. This concept can be seen as a 3D model of 2D geographic information with scale factor as the 3rd dimension. This is the formation concept of SSC (Meijers and van Oosterom 2011). Simply speaking, SSC is a smoothed tGAP. The main advantage of extruding 2D geoinformation to the 3rd dimension is to

smoothen inter-LoD changes as it was discussed before. Assuming the 3D data structure, it can derive geoinformation visualization by performing an intersection between an imaginary plane with the dataset (this operation is called slicing). Moving this imaginary plane on the scale axis would change the scale of the derived data. Primary slices have a plane perpendicular to scale axis and parallel to x-y geoinformation plane. There would be also possibility of deriving slices with planes not parallel to the x-y geoinformation plane. An imaginary SSC is shown in figure 2.7. Comparison between a smoothed

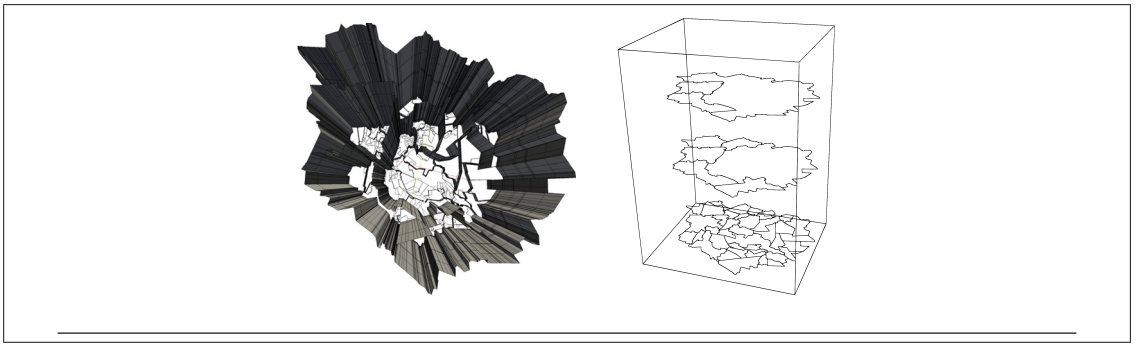

FIGURE 2.7: Inside an imaginary SSC. From a top view (left) and several cross sections (right). Figure taken from Meijers and van Oosterom (2011).

tGAP and a classic tGAP is shown in figures 2.8 and 2.9. A common important part

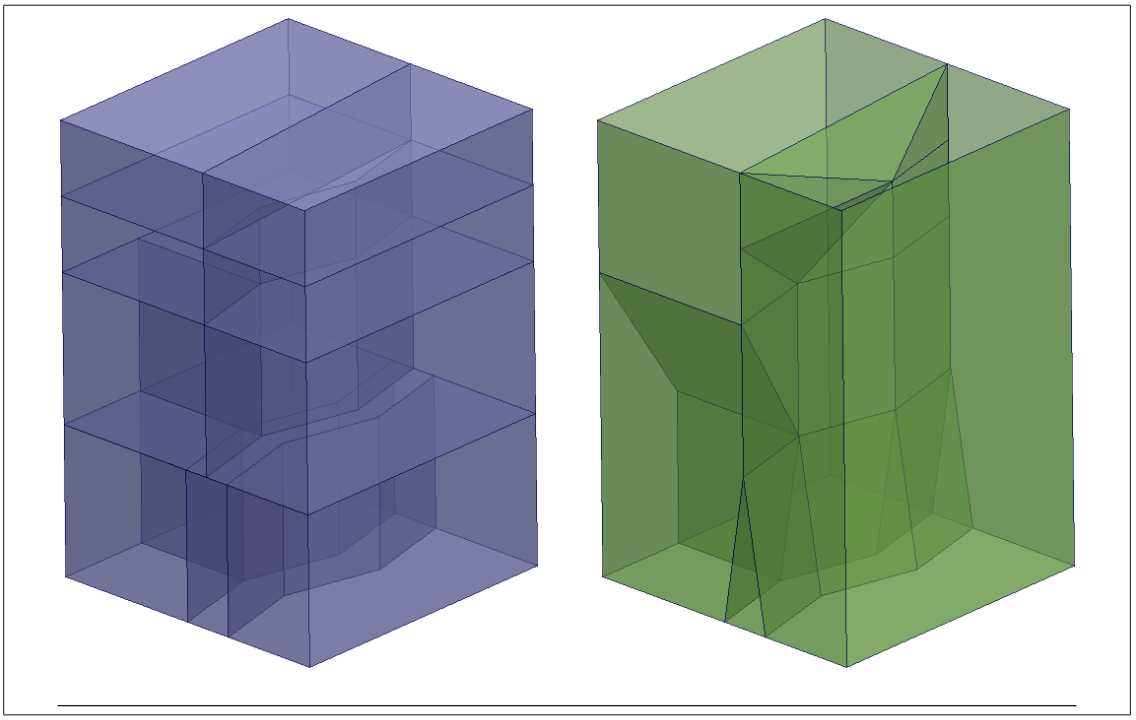

FIGURE 2.8: Classic tGAP (left) and the smooth tGAP (right) based on an imaginary dataset. Pay attention to the smoothing between the LoDs. Figure taken from (Van Oosterom and Meijers 2011).

of GAP, tGAP and SSC is the Binary Line Generalization (BLG) tree (Van Oosterom 2005). This tree is the structure responsible for cartographic line simplification. The

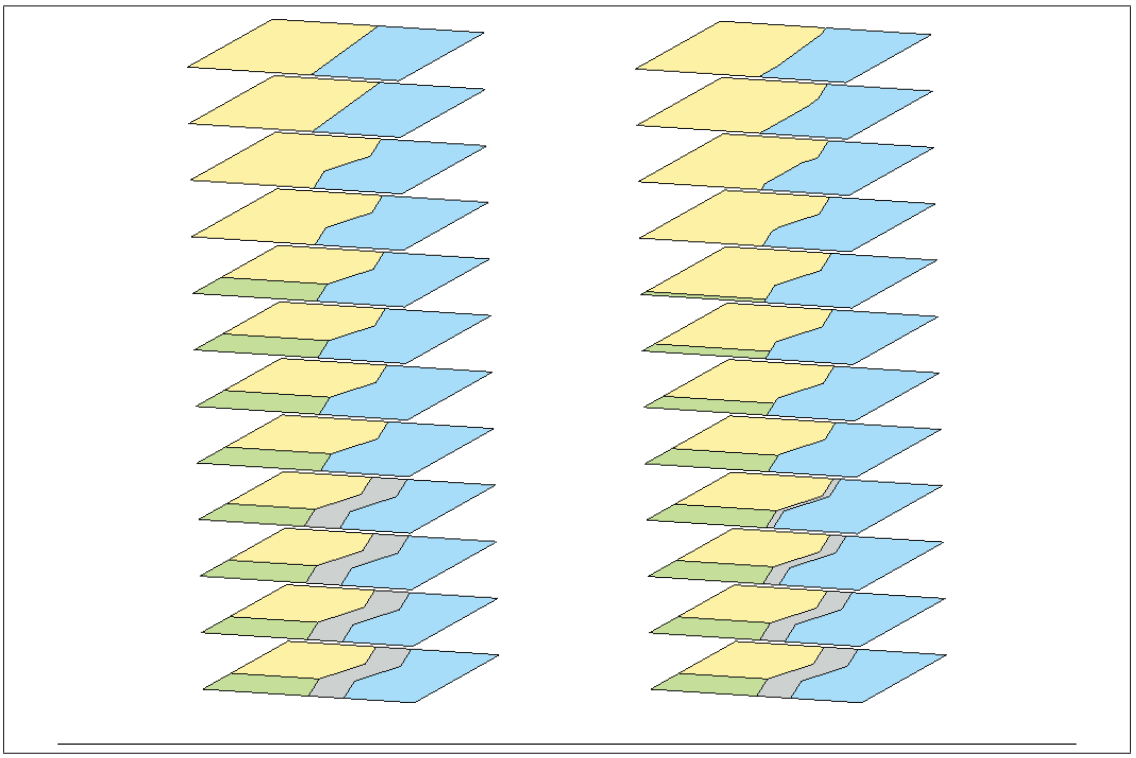

FIGURE 2.9: Comparison between classic tGAP (left) and the smooth tGAP (right) slicing results based on an imaginary dataset. Figure taken from (Van Oosterom and Meijers 2011).

simplification is based on Douglas-Peucker line simplification algorithm (Douglas and Peucker 1973), but is a storage algorithm for faster retrieval of data. BLG (abbreviated term for Binary Line Generalization) is a binary tree in which every node saves its tolerance value from the algorithm. The higher the tolerance value, the higher the node will be in the BLG tree. The leaves of each tree node points to the next higher tolerance values after that node, one between node and segment start and the other between the node and segment end. The concept is shown in figure 2.10. As it was mentioned, the main usage of BLG is to traverse the tree to build up edge segment from lower to higher scale (from root to leaves traversing).

2.4 Efforts on Generalization of Road Networks and Similar Other Networks

Networks, whether being urban roads, hydrologic, pedestrian paths, human-made waterways or other type has importance in GIS and cartography. Modeling spatial information of networks in computer systems also contains the essential phase of generalization. Generalization efforts on networks can be seen from two points of view:

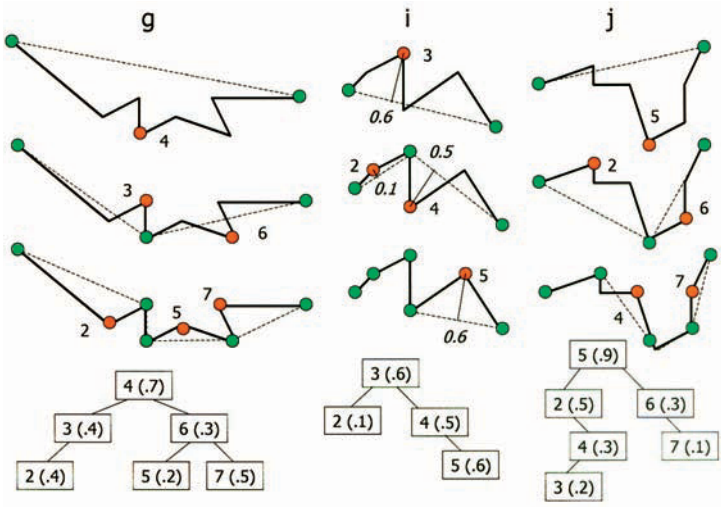

FIGURE 2.10: BLG tree formation and comparison with the line geometry of three edges. Figure taken from (Van Oosterom 2005).

- Geometric generalization: This category contains the variety of processes who are defined to simplify and smoothen the geometry of the networks features. Criterion of such processes can be based on geometry itself, semantics or structural aspects of the feature.
- Modeling generalization: Efforts who mainly consider selection, omission and abstraction of the network based on one or more semantical and/or structural characteristic of the feature(s).

2.4.1 Geometric Generalization of Networks

There have been and still are efforts on generalizing the individual features of a network. Mostly the efforts are on simplification of the geometry or to smooth the feature. Several algorithms will be mentioned here.

- **Douglas-Peucker:** Introduced by Douglas and Peucker (1973), simplifies the line segments by iterative omitting of points with less importance. Importance is evaluated by distance from intermediate and the point itself, comparing it with a tolerance.
- **Visvalingam-Williamson:** Introduced by Visvalingam and Williamson (1995) and is basically similar to Douglas-Peucker in simplification and evaluating importance. The difference is on the importance measure. In this algorithm, the area

of the triangle shaped by the point and two neighboring points is considered as measure. The lower the area, the less important the point.

- **Morphing:** Based on Nöllenburg et al. (2008), this is a morphing algorithm supporting continuous generalization. The algorithm relies on finding optimum correspondence between two different polylines (in source and goal scales visualization). After finding such correspondence, the transition would be possible for any stage between the two levels. The decision is made based on several geometric and semantic properties of the two polyline segments.

2.4.2 Model Generalization of Networks

The generalization of networks used to be thought of indifferently in comparison with other geographical data types. Researchers in field of cartography have tried to understand and model the characteristics of networks. Two similar and important outcomes are Horton's and Strahler's stream ordering principles by Strahler (1960) and Horton (1945) with a slight difference. These principles suggest an ordering process of streams with a directional hierarchy in mind. The order would help to select higher order streams in case of need for selection (as it is needed for selection operator). A comparison between two algorithms is shown in figure 2.11.

FIGURE 2.11: Comparison between Horton (left) and Strahler (right) principles of ordering streams. Figure taken from (Cecconi 2003).

Later efforts mostly consider modeling networks using graph theory capabilities since the previous methods did not consider enough information from the network structure itself rather than considering the representational aspects of the network. As both water (hydrology) networks and road networks are both of interest of researchers in this field, Haggett (1967) tried to transform previously mentioned methods to road networks. Later initiated by Keates (1989), there have also been feature classification methods who

use graph aspects to classify features for generalization and graphical representation by means of quantifying relative importance of individual segments of the network. These quantifying characteristics normally include node and edge degree measures. (Mackaness and Beard 1993; Thomson and Richardson 1995) suggested use of more complex graph algorithms (Kruskal's and Shortest Path Spanning Tree algorithms respectively). For an example of Thomson and Richardson algorithm refer to figure 2.12. While classifying

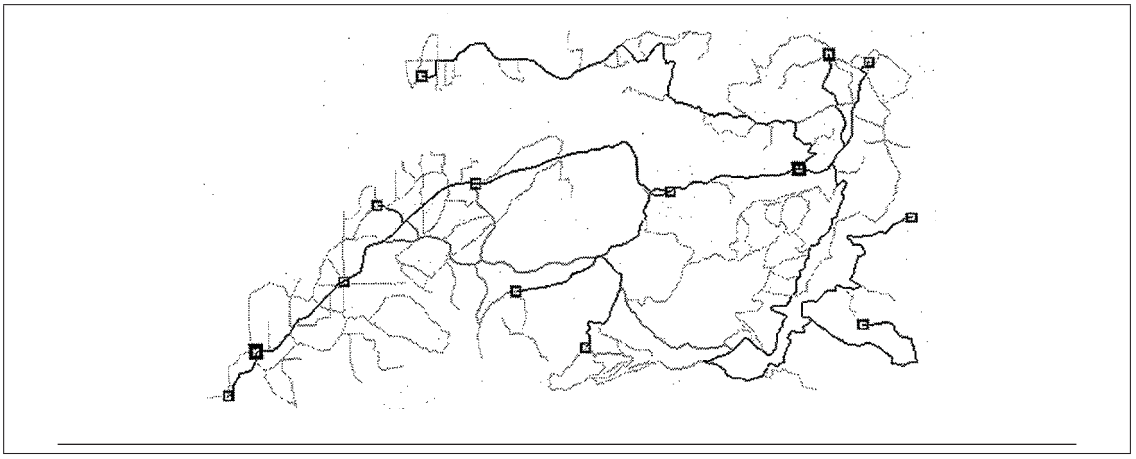

FIGURE 2.12: An example of the algorithm from Thomson and Richardson (1995). Selection of several road feature based on Shortest Path Spanning Tree algorithm, considering the feature length as cost value. Figure taken from Thomson and Richardson (1995).

road segments, in order to treat segments as one meaningful continuous entity, there would be need to identify a set of one or more segments who are semantically one entity (but have been modeled by several segments). This common entity is titled "Stroke" by Thomson and Richardson (1999). The discussion is on modeling coinciding segments with closest degree to 180^{o} to support the continuity concept. Such segments would be considered to be the same street segment under certain conditions.

As modeling of networks to graphs have been mentioned, most of the research works have tried to model the network to a straight-forward model of edges and nodes. Transforming the street network to the graph would mostly convert junctions/intersections to nodes and the connections (segments) to edges. A dual graph (named "connectivity graph") which exchanges the meaning of nodes and edges have been introduced by Jiang and Claramunt (2004). In this concept the network intersections are modeled by graph edges and road segments are modeled to graph nodes. The new data model makes new potential analysis reachable. An example is shown in figure 2.13.

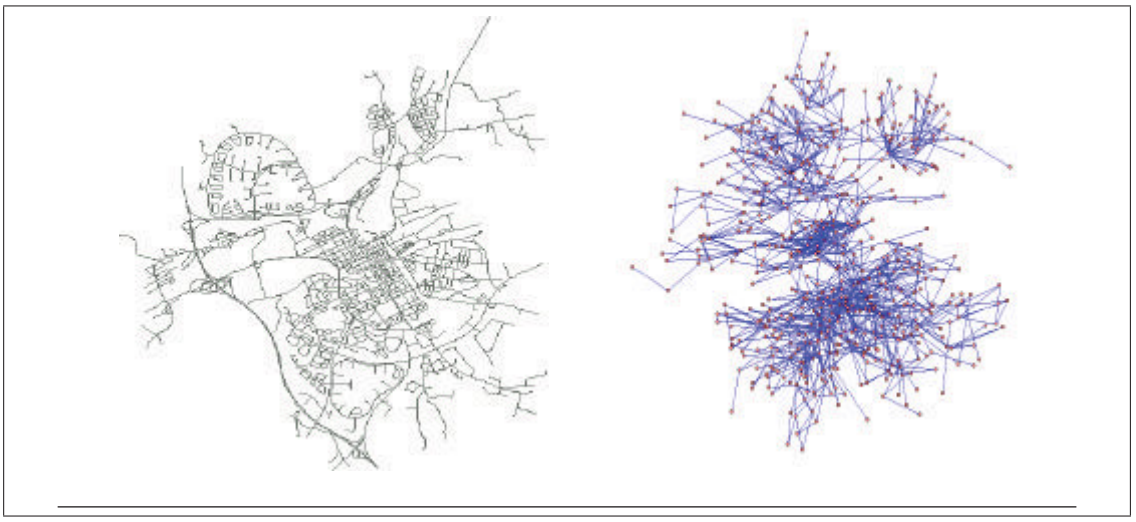

FIGURE 2.13: Connectivity graph concept. Road network of city of Gävle, Sweden is shown as a network (left) and as connectivity graph (right). Figure taken from Jiang and Claramunt (2004).

2.5 Clarification of Interest

Summing up the background, the interest area of this research work should be clarified. This research work can be seen more as a *model generalization* effort which results in a *vario-scale generalization* method. Considering generalization operators, the interest is around using *selection*, *omission* and *simplification*. And it is based on aspects of road networks data modeling mostly from graph theory point of view.

Chapter 3

Theory of Road Network SSC

In order to import and query the data structure which is point of interest of this research work, there is need to adapt the the data to the specification and constraints of such data structure. This process needs to have understanding of both sides: *data* and *data structure*. The discussion in this chapter will be around data, preprocesses and processes of road network SSC from a theoretical point of view. In case of need the discussion provides required background.

3.1 Background of an SSC

As it was discussed before in section 2.3, an implementation of an SSC relies on the tGAP data structure. According to Meijers and Van Oosterom (2011), SSC is a smoothed tGAP which supports the smooth zooming between different LoDs. This means that in order to implement or develop SSC concept, a potential solution should implement a structure similar to tGAP (which is a combination of tree structures itself), afterwards the solution should adapt the structure to support smooth zooming.

3.1.1 tGAP

tGAP is composed of a tree describing the hierarchical relation of the area features and a set of trees, so called *forest* who keep the parent-child relation between edges. In the original tGAP formation, the interest is to abstract and generalize area features. A *face-tree* stores the realtions between the area features and the *forest* stores the edges, which are the border definition of topology of area features (neighborhood, connectivity and etc.). The hierarchy is based on activity of two generalization operators: *eliminate*

and *merge*. In order to provide continuous generalization, a continuous simplification is also necessary.
Building up the data structure is an iterative process. In every iteration the least important feature of the dataset is being selected as an elimination candidate. The decision criteria is based on feature class and the area occupied by the feature (but in theory the criteria can be another quantity measure). The lower the measure, the less importance the feature is assumed to be (and vice versa). Beside having the elimination candidate, there should also be a merge candidate (to whom the area of the eliminated candidate would be assigned to). The decision of the assigning area from the less important feature to the higher important feature(s) can be done in two suggested ways:

- Evaluate the merge candidate as having the highest compatibility (based on shared border length and class compatibility), as mentioned in Van Oosterom (2005).
- The eliminate candidate area would be divided to two or more parts and it would be assigned to the appropriate neighbors along the border side, as mentioned in Meijers and Van Oosterom (2011).

After performing the elimination and merge operations, the hierarchy relation is being assumed as the eliminated feature to be child (leaf) and the feature which receives new area is assumed to be parent (root). The iterations continues to result in one area feature covering union of all the initial area features. The result is being saved in a tree structure (face tree) in which every tree node contains the importance value of the respective feature (one step of the process is being shown in figure 2.5). There is also need to apply similar process on the boundary edges of the area features. In the case of edges, the elimination is not independent process and the candidates are selected based on the face they circumscribe. The edges survive when areas on their both sides (left-right topology) survive in the elimination/merge process. In case of eliminaiton/merge process on any of the neighboring areas, the edge may be omitted or be merged with several other edges. In case of edge merge, a parent-child relation would be built and stored in a tree structure. But in case of omission of an edge, such tree would be complete and will not grow more. This phenomena is the reason of having multiple hierarchical trees for edges which are called a *forest* altogether. Figure 2.6 illustrates an edge forest.
To query the data structure, there would be an importance value as the query parameter. Having features hierarchies, querying the data structure becomes possible. If importance value criteria of certain area feature covers the query value, that feature would be included in the result set, if not the traversing can continue by visiting feature's parent or children.

3.1.2 Smoothing tGAP

While forming the tGAP data structure, with every iteration one LoD is being abstracted and stored. The idea behind SSC is to extrude ordinary *x-y* geoinformation to 3rd dimension with scale as the new dimension. As mentioned in Van Oosterom and Meijers (2011), at query and visualization time the transition between these LoDs (which is equivalent to moving in the 3rd dimension) results in visual sudden changes or jumps. This inter-LoD jumps should be replaced by a smooth effect. The solution should give smooth transitions for inter-LoD query values. So for any small change in the scale query parameter (or importance value parameter) there should be a small change in the map content. As the LoDs formation is naturally a discrete process, the smoothing solution should smoothly *connect* between these states (which is not the case in the original tGAP structure). As this generalization process is composed of set of operators, each individual operator should support continuous smoothing between LoDs. The active operators are elimination/merge and simplification. Merge and elimination act concurrently as one feature is being eliminated and the orphaned area is inherited by the neighboring candidate feature. Continuous smoothing of this phenomena is modeled to shrinking for the eliminated feature and growing for the merged feature as mentioned in Van Oosterom and Meijers (2011). In a 3D view, moving toward higher values of scale axis (equal to smaller map scale and higher importance values) on one hand the borders of the eliminated feature is moving inside toward the area center and results to area decrease. On the other hand the feature who inherits the area is growing in area by expanding the borders to the center of the eliminated feature. The process continues to the point where the eliminated feature become a singe point and later disappears and the inheriting feature takes the whole orphaned area (the process can be seen in figure 2.8 (right) and the effect of smoothing in slicing result can be seen in figure 2.9 (right)). It was mentioned before that the simplification operator result is being stored in a BLG tree. From 2D point of view the operator is a Douglas-Peucker simplification operator (Douglas and Peucker 1973) in which less important points are being omitted according to a specific tolerance value and the process of node elimination continues and give a node sequence at the end. Performing the process in 3D formation is shown in figure 3.1, where the left illustration shows an un-smoothed version and the right illustration shows the smoothed version. In order to implement such smoothing in 3D, three triangles are being formed to smoothen omission of an individual node in a continuous geometric way. Smoothing is achieved after definition and implementation of generalization operators in a smooth way to act on the 3D setup.

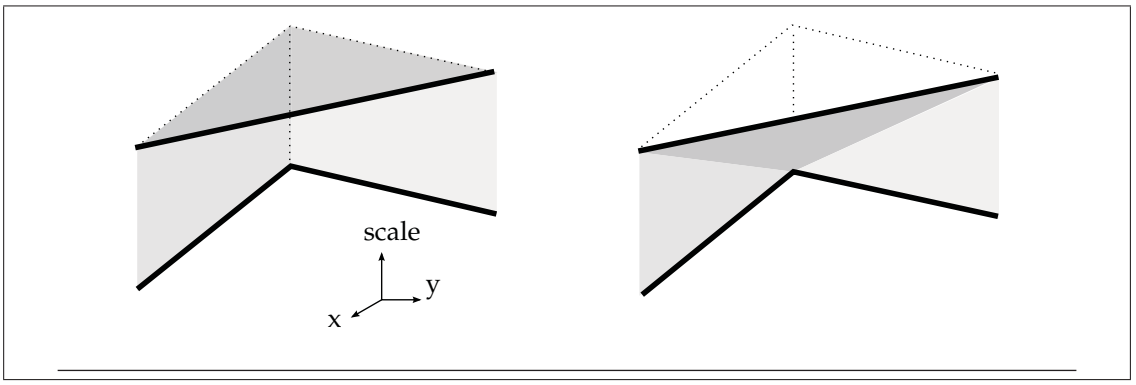

FIGURE 3.1: Line simplification in 3D, in which a point is being omitted from the line. On the left formation, the omission is modeled to 2 rectangles and 1 triangle. Moving to higher values on the scale axis will result in abrupt changes. On the right the smooth solution is given which is modeled by 3 triangles. The transition from a corner to the segment is mainly modeled by the triangle in the middle. Figure and description from Van Oosterom and Meijers (2011)

3.2 Road Network as a 'Network'

As it have been mentioned before in section 2.4.2, the topology relations of road networks are formulated by connections between nodes, node neighborhood, edge neighborhood, edge intersection, paths and etc. Several researchers have concluded that graph theory is a promising way to reflect the connectivity-based nature of a network. With a model based on the graph theory, the road segments and junctions would be modeled to edges and nodes respectively. This would help to analyze the network but if there are needs to visualize the road network, there should also be means to store the geometry of the network.

Having the correspondent graph of a road network, it would be simple to store and access the network information. The graph G would have set of nodes V and set of edges E which is formulated as $G\{V,E\}$. Considering the network as a graph, two points should be mentioned here:

- The model being used should be able to store semantic data related to road segments and junctions in their graph correspondents (edges and node respectively). This means that graph edges should have the potential to store and attach data such as road segment name, class, width and etc. Beside edges, the storage should be able to store and attach information such as junction name to nodes.

- Beside holding semantic information and connectivity information, road segments have geometry. Normally such information is formulated by sequence of points; which form the road segment geometry when being connected from the first one to the last one. On the other hand a dataset storing a graph (like an adjacency

matrix), normally stores nodes and the connections between them. The connection between two points/nodes can be simple, directed or multi but in any case the topological definition of the connection describes the start and the end point. This means that normally only the first and last point of the road segment geometry are being saved as graph vertices. The incompatibility between segment geometry and graph topology connectivity should be handled in any road network storage system. The detail of the implemented solution on this problem for the case of this research work is discussed in the next chapter.

3.2.1 Short Background on Graph Theory

Several definitions about graph theory are given here which are going to be used later (definitions from Beck, Bleicher and Crowe (2000) and Chris Caldwell's "Graph Theory Glossary"[51]):

Graph: a simple graph is a (usually finite) set of vertices or nodes *(V)* and set of unordered pairs of distinct elements of V called edges *(E)*. Graphs can be simple, directed and multi. In simple graphs, there would be maximum one edge between two vertices/nodes. Directed graphs or digraphs have edges who have directions. Multigraphs potentially can have more than one edge between a pair of nodes/vertices.

Degree: The degree (or valence) of a vertex (or node) is the number of edge ends at that vertex.

Path: A path is a sequence of consecutive edges in a graph and the length of the path is the number of edges traversed.

Connected graph: A graph is connected if there is a path connecting every pair of its vertices/nodes. Testing connectivity of the graph means checking whether the graph is connected to not.

Beside the given basic definitions, two more concepts which are not included in classic graph theory should be mentioned. *Shortest path* between two nodes is a path in a graph which minimizes the sum of weights of visited edges in that path. This definition applies to graph in which the edges are given weight values. If applied to graphs without weighted edges (which means that the weight would be equal for any edge), the algorithms gives the path which passes throughout the least number of nodes/vertices. Shortest path is used in definition of *edge betweenness centrality*. According to Brandes (2001), betweenness centrality of an edge is the sum of the fraction of all-pairs shortest paths that pass through it.

3.3 Formation of Road Network SSC

Implementation of road network SSC is based on several necessary processing steps of the initial data. The initial data of the road network in SSC contains the geometry and semantics of the network. The data contains geometry of road segments which is normally available in datasets from any data provider (but potentially with different formatting); besides the geometry there would be several arbitrary semantics of data which is normally defined by the data provider. As it will be discussed later in this chapter and the next chapter, semantics of road segments could potentially be used in order to set importance values of road segments. The following sections discuss the process of formation of the data structure.

3.3.1 Geometry

In order to describe line geometry, normally dataset is composed of several polylines (or line strings). Each polyline would be composed of sequence of points (nodes). This means that by having nodes (who have x-y coordinates) and keeping the sequence of them in a polyline, it would be possible to describe the geometry of that individual polyline, as in (OpenGIS Implementation Specification for Geographic information - Simple feature access - Part 1: Common architecture)[52]. In the road network SSC dataset, the final storages is a RDBMS (short form for Relational Database Management System) with spatial capabilities in order to provide possibility of efficient data access and potential future use from other systems. Therefore the initial data should be read, processed and saved in an RDBMS. Two different entity sets will be stored in the data structure. One set would store the data of segments (edges) and the other set would store the data of points (nodes). By having the coordinate-aware individual nodes and their sequence order, it is possible to form the line segment (edge) geometry at retrieval time. So the only geometry being saved in the data structure are the coordinates of nodes. There would be a third entity set being stored in the data structure which describes the inclusion relation between nodes and edges (this relationship is also being used to fetch appropriate nodes for segment simplification). Inclusion relation would enable the structure to fetch nodes of a segment to build up its geometry (by queueing the nodes according to their sequence). A primary processing of data would be fetching edges and nodes and their relation (which nodes are included in which edge?) from the input data. Two points should be mentioned here about the storage of geometry:

- Beside storing the inclusion relation between nodes and edges, the **order** of the nodes in the sequence is saved in the same storage. Further detail is discussed in the next chapter.

- In case of having intersection of two or more segments, the intersection node would be included in the sequence of two or more segments. So join set of that node would contain more than one segment (edge).

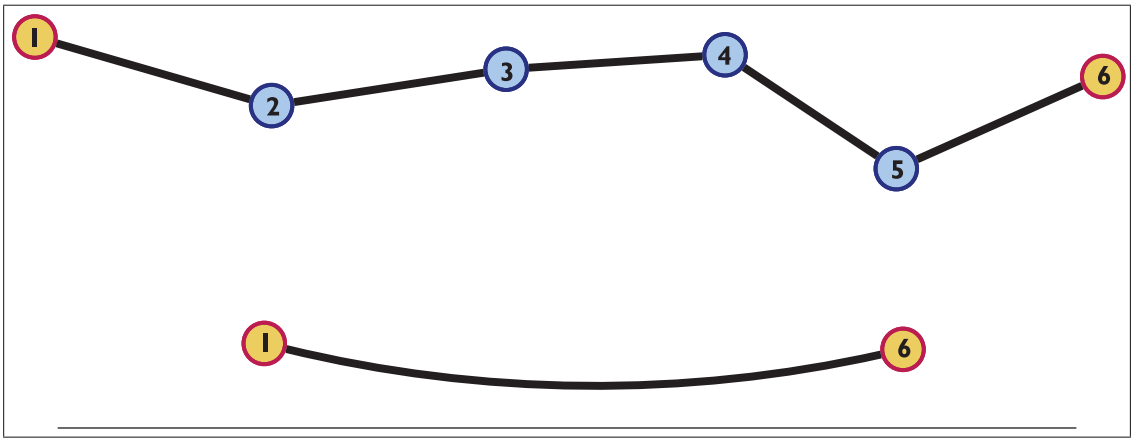

FIGURE 3.2: An imaginary segment is being shown on the top and the graph equivalent is show on the bottom. All of the points will be saved to the data store with their coordinates and their sequence. But only 1 and 6 would be added to the graph vertex set. The connecting edge would also be added to the edges set of the graph.

3.3.2 Network Topology

In order to derive the topological connectivity information, and beside the geometry processing (which was discussed in section 3.3.1), a temporary graph data structure is used to save the network topology for analysis purposes. The topological relationship of the features are formulated in the form of connectivity of nodes (neighborhood, paths, distance and etc.). The potential data structure will need means to describe two main entities: *edges* and *nodes*. As it has been discussed before in section 2.4.2 graphs are appropriate data structure for modeling road networks topology. Topological connectivity of the network is not described in the input data structure explicitly but such information is rather described implicitly. In order to form the graph correspondent of the network, in the first step an empty graph would be created in the memory. Then an iterator will iterate on segments individually. Iterating over each segment includes also iteration over its points. Accessing segment points, the first and last points of each of the segments will be added to the graph and the segment itself would also be added as an edge. Figure 3.2 shows the mentioned process. Several aspects of the graph would be used in the process, namely *shortest path*, *connectivity* and *edge betweenness centrality* which already have been covered in section 3.2.1.

3.3.3 Building up tGAP on The Road Network

The tGAP for the road network initially needs a set of geometries (as it has been discussed in section 2.3) and an importance measure. This measure would be used to differentiate more important features from less important features. The idea behind the the structure is to keep the more important features and eliminate less important feature for better abstraction and generalization. The importance measure can be from any geometric, topologic, semantic or arbitrary characteristic. Quantities and measures who give unique values for individual segments are of higher interest. This would result in more efficient ordering of the segments. As the original tGAP uses mixture of feature class and area as importance evaluation (Van Oosterom 2005), a product of length of road features and its feature class can be a measure for segment importance evaluation. There could be some other suggested measures as connectivity graph degree suggested by Jiang and Claramunt (2004), road class, distance from network center and etc. Edge betweenness centrality has been discussed in section 3.2.1, this measure can reflect edge criticality in the road network. If the segments importance values are available, the formation process of road network tGAP can take place.
The remaining text assumes that four data objects are available: A *graph*, a *dataset*, a *forest* and an *array*. Initially the *graph* is an empty graph (empty node set *(V)* and edge set *(E)*) and the *forest* is also an empty forest. A forest is a set of trees, each tree can have one or more nodes. The *dataset* contains the processed information of road segments stored as edges, nodes (points) and their entity relationship. As it was discussed before, basically the dataset is a spatially-enabled RDBMS. The *array* contains two columns, first column holds edges and second column holds importance value of that edge; so each record of this *array* stores a (edge , importance_value) pair. The descriptive process of building the road network tGAP is as follows:
First the array is being iterated and then it will be ascendingly ordered based on the importance values and the values themselves will be normalized to [0,1] range. Now the *graph* is being formed using an iterator which goes over the edges of the *dataset*. In every iteration :

- First node (point) of the edge is being fetched.
- Fetched first node is added to the node set of *graph*.
- Last node (point) of the edge is being fetched.
- Fetched last node is added to the node set of *graph*.
- A new edge connecting the mentioned first and last nodes are added to the edge set of the *graph*.

After iterating over the segments of the *dataset*, the *graph* is formed with the whole topology of the road network. At this moment a process of edge and node elimination would take place. Weeding the *graph* will result to formation of the *forest*. The *forest* (which is the tGAP edge forest) will contain the parent-child relation of the edges in order to prioritize their representation in the generalization process. Ideally the *forest* should be stored in a separate storage but in the implementation, the data of the *forest* will be included in the *dataset*. Further discussion of this idea is done in the next chapter. Therefore for better understanding of the algorithm the term *forest* will be used in this chapter to relate to the structure but it can be thought of as a mixture of *dataset* and *forest* from implementation point of view. The *array* holds the ordered list of edges and will acts as a queue. The elimination is performed on the *graph* edges and nodes and the results will be saved in the *forest*. This means that by the end of this process, the *graph* will be empty and all of the road segments would be modeled in the *forest*. The process is again an iterational process. In every iteration:

- The edge of the *graph* which is correspondent to the first item of the *array* (meaning to be the least important feature) is being selected as elimination candidate.
- Does elimination of the candidate result in *graph*'s violation of connectivity?
  - Yes: Insert the edge at the end of the *array*. Pass to next iteration.
  - No: Delete the edge (from the *graph* and the *array*). Decide on the nodes of the deleted edge. On any node, consider the node degree (valence):
    - 0: Delete node.
    - 1: Pass to next iteration
    - 2: Do the following actions:
      - Delete the node (from the *graph*).
      - Delete the edges of this node (from the *graph* and the *array*).
      - Add a new edge connecting two ends of the deleted features (to the *graph*).
      - Set the new edge as the deleted edges' parent (in the *forest*). This means that the new edge would be a root to a tree and the deleted edges would be its children.
      - Add the new edge and its importance value (maximum of its children importance values) to the *array*.
    - More than 2: Pass to next iteration

There are some descriptions necessary for the process. First of all as it is has been seen, it is important that in every iteration the the network does not become a disconnected

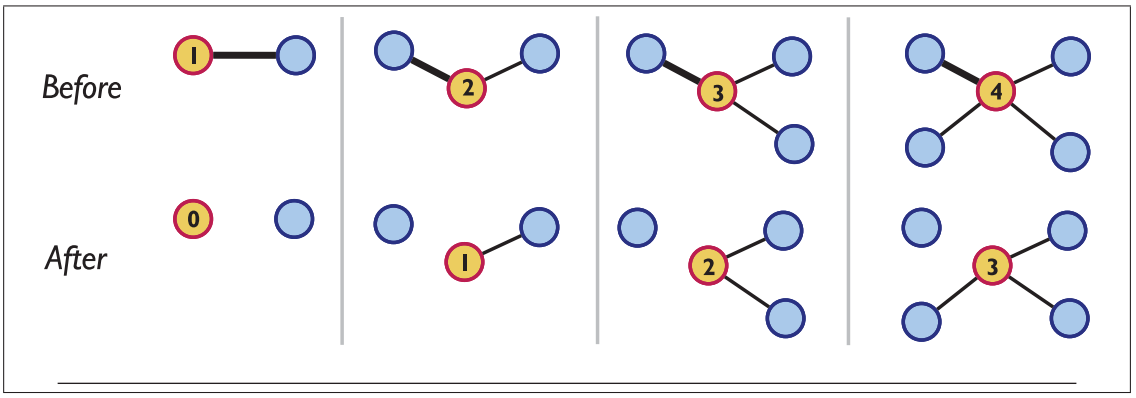

FIGURE 3.3: Four situations are shown before and after edge elimination. Every column shows an example of one situation. The top row shows the formation before the edge elimination and the bottom row shows the formation after the elimination. The nodes of interest are shown in orange and the elimination candidate edges are shown in thicker format. Numbers show the degree of the node.

network. By this means in every iteration a connected network is available, so later at query time by reconstructing the situation through querying the data structure, the result would be a connected legitimate network. A network being disconnected means that there would be two sets of nodes of which one or more pair of them have no possible path connecting them. A common case is that by removing one edge, the result is a lonely node with degree 0 on one hand and the remaining *graph* on the other hand. In this case, the structure remains connected by eliminating the lonely node.

Another point which should be discussed is the decision made on the degree (valence) of nodes. The general idea behind this operation, is the interest of building a hierarchical data structure with regard to connectivity topology. Because of the complexity of the networks, it is hard or impossible to form such relation in the beginning. But this process weeds the structure by eliminating the less important edges to simplify the network and make it more abstractable. By iterating through this process, number of edges and nodes and logically number of connections between them decreases and thefore network becomes less complex. While iterating over network edges there would be situations which a node has degree of 2. Such node has just a role of connecting two edges and adds no more to the connectivity of the network. There is a possibility to delete that node and eliminate two edges simultaneously. Then an edge would replace this structure to keep the connectivity. The replacement edge is a more abstract structure which represents the two deleted edges and is appropriate for higher LoDs. This is thought of as a child-parent relation as the new edge (being higher in the hierarchical structure) becomes the parent and the edges with lower hierarchy are the children (deleted edges). This set of relation will be saved in the *forest* in order to fetch the appropriate edge based on the queried value. Different cases of node degree is shown in figure 3.3 and a sample process is shown in figure 3.4. More detail on implementation of this idea is discussed in the next chapter.

Another aspect is about the importance values. As it is mentioned in Van Oosterom

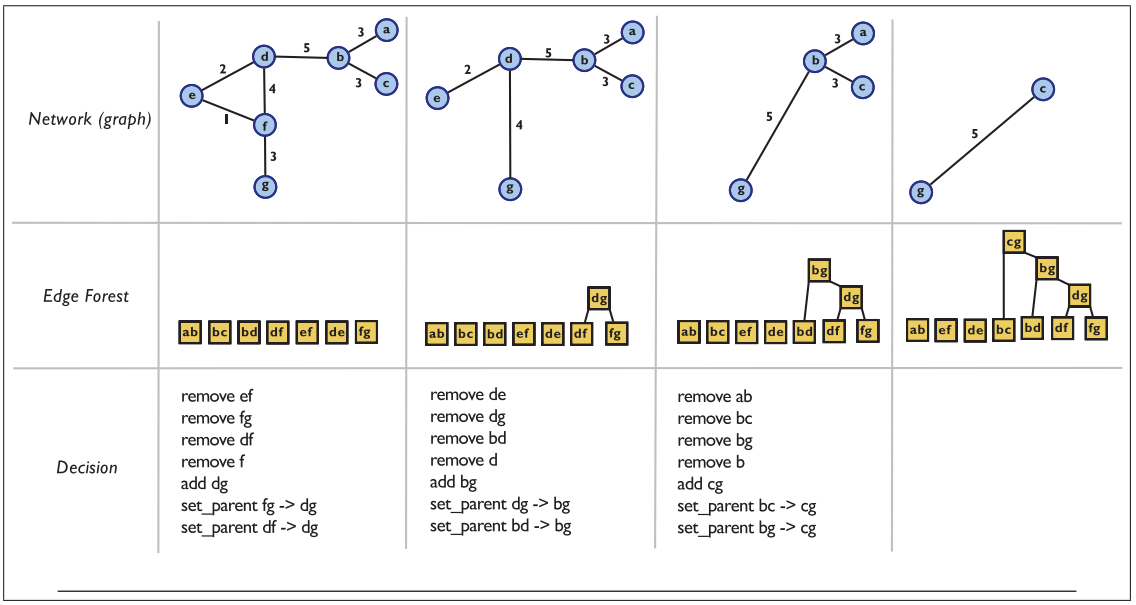

FIGURE 3.4: A part of an imaginary network is being shown. The network is shown as a graph on top row. Its equivalent edge forest is shown on the middle row. Finally the bottom row shows the decisions on each step. Four steps are being shown. Nodes are labeled with one letters. Edges are labeled with two letters. The numbers show the importance values. In this case the values have been derived through betweenness centrality analysis, but the numbers have been exaggerated and rounded for demonstration purposes. Decisions are made based on importance values and degree of nodes.

(2005), there are two importance values which define the feature: a *minimum* and a *maximum*. At query time, the edge is being fetched if the query parameter is between minimum and maximum. As it was discussed earlier in this section at the beginning of the *forest* formation, single edges would be assumed as single roots in the forest setup. At that stage, the minimum importance value for all of the edges will be set to 0 and the maximum importance value will be set to the edge importance value from the *array* record corresponding to that edge. While iterating over the structure and in case a new edge is added, the minimum value of the new edge would be set to the maximum of its children's maximum importance values. By these means, every edge will have minimum and maximum importance value in the *forest* and the criteria also will control no overlapping importance value for parents and children. After finishing the iteration over the edges and weeding the *graph*, the *forest* will be ready containing several trees who can be single node trees or combination of root and leaves.

3.3.4 Smoothing of Road Network SSC

As it was mentioned earlier in section 2.3 and based on suggestions from Van Kreveld (2001), there should be a strategy which provides continuous effects for the generalization operators. In the case of road network SSC, the operators are *elimination*/*merge*

and *simplification*. The strategy for each operator is being discussed in the upcoming sections.

3.3.4.1 Smoothing Elimination

Elimination is the process of the feature being removed from the graphical representation. According to Van Kreveld (2001), in order to manipulate such effect in a continuous way there would be possibility to implement a *fade out* effect. In computer graphics, such effect can be performed by changing the transparency of the feature visualization. This means that a continuous effect can map the inter-LoD parameter values to transparency of a range between fully opaque to a fully transparent representation.

3.3.4.2 Smoothing Simplification

The line simplification with a continuous implementation should also be considered. Two options can be considered:

- **3D solution:** According to Van Oosterom and Meijers (2011) and as it was mentioned before in this chapter, the Douglas-Peucker line simplification algorithm (Douglas and Peucker 1973) can continuously be implemented in a 3D setup. The process is done by repeating the line in the 3rd dimension while eliminating one node in the new level (e.g. level n+1). The correspondent node in level n should be connected to neighbors of that node in the level n+1 to form a triangle. Two more triangles will be formed by connecting that corresponding node in level n to neighbor nodes in level n and level n+1. The solution is shown in figure 3.1. While this solution is theoretically correct, a semantic issue is applied to it. Deriving inter-LoD simplified lines give out results which are not simpler. An example is shown in figure 3.5.

- **Pseudo-3D solution:** Another possibility to implement a continuous line simplification is a Pseudo-3D solution. Assuming two levels (n and n+1), a point is going to be eliminated. The position of this point at level n is connected to the mid-point of the line segment connecting its neighbors at level n+1. This virtual new line segment is the displacement of the point between level n and n+1 (after n+1 the point would be no longer existent). Any inter-LoD parameter value which is between n and n+1, would be mapped to the new virtual line segment and the new position of the point is being determined by such mapping. Since the movement is a linear process with known start and end, the inter-LoD could be easily mapped to this line segment. This solution does not have a real 3D equivalent,

therefore non-planar slices of the SSC based on it would not be possible. On the other hand the line simplification leads to a less simplified representation of the line segment without any further high impact on the process. The process is shown in figure 3.6.

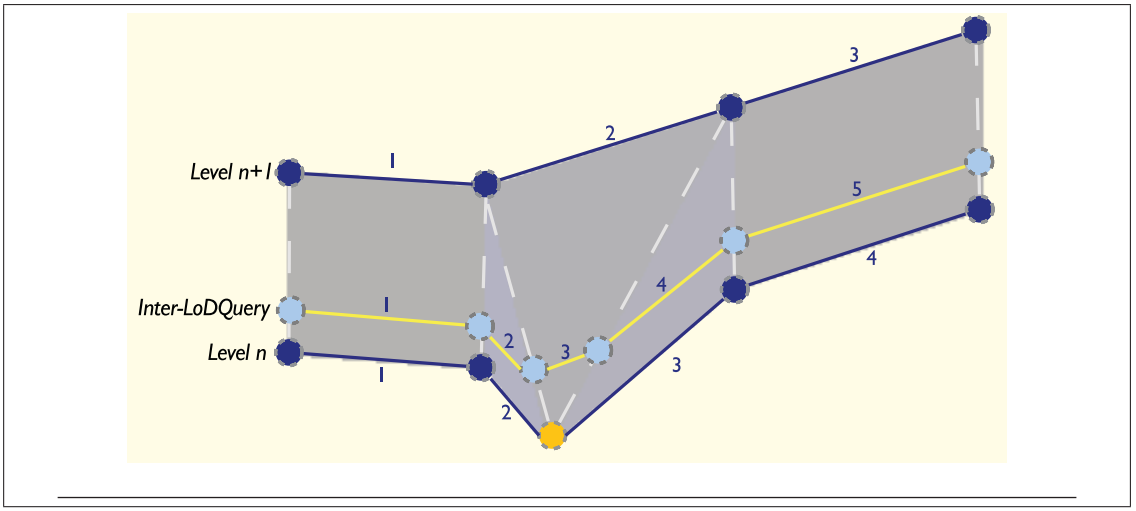

FIGURE 3.5: The continuous line simplification in a 3D setup is being shown for an imaginary segment in two levels. In the lower level (n), 5 nodes and 4 edges are being shown while the orange node is being eliminated in this iteration. Thus the next level (n+1) contains 4 nodes and 3 edges. Continuous simplification is implemented by 3 marked triangles in the middle. An inter-LoD query is shown (in yellow), who queries the structure between n and n+1 levels. As it can be seen the resulting simplified segment is visually and geometrically more complex than the original segment (having 6 nodes and 5 edges).

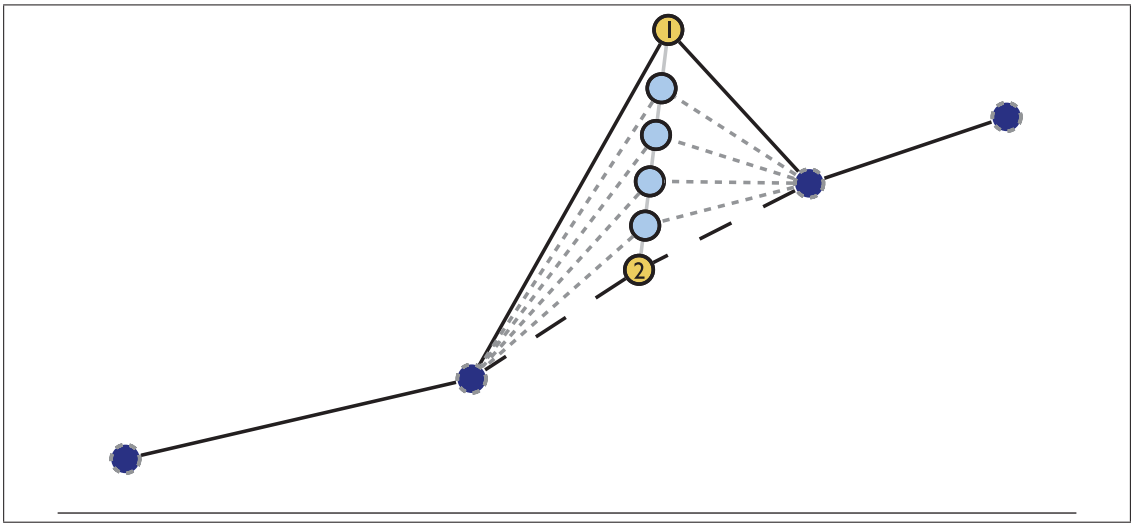

FIGURE 3.6: The Pseudo-3D approach is being shown. An imaginary segment is being shown in 2 levels. Point 1 should be eliminated. In this approach the continuous simplification is done by moving the point to the mid-point of the segment connecting its neighbors (movement from 1 to 2). For inter-LoD queries, the point position is found through mapping of query parameter to length of line segment connecting 1 and 2. Light blue points demonstrate some potential inter-LoD queries.

3.4 Reading from a road network SSC

Having the road network tGAP with implemented smoothing, the data structure is query-ready. Basically querying of the data structure is sending a query parameter to fetch appropriate features from the *dataset*. After getting the result set, they would be visualized to the potential user.

3.4.1 Discussion on Scale

Importance values play a crucial role in formation of the SSC and retrieval of geoinformation through this method. The presence of features on the final product is based on their importance values. Querying the *dataset* is basically possible by having an importance value as the query parameter. But as the product of the data structure is a cartographic product, the data retrieval should be parametrized with scale. According to radical law of Töpfer which was introduced in Töpfer and Pillewizer (1966), four quantities play role in transformation of scales. Radical law is being shown in equation 3.1:

$$n_f = n_a \cdot \sqrt{\frac{M_a}{M_f}} \tag{3.1}$$

where n_a is number of features in the source map, M_a is the scale of the source map, M_f is the scale of the derived map and n_f is the number of the features in the derived map. By having a source map, the quantities regarding the source map are known. In SSC data structure, importance value determines the presence of features in the result map. In order to calculate the scale of the derived map, there is need to transform importance values to number of features. So by having an importance value, number of features would be calculated and then the scale is calculable.

There could be another approach which assumes that the scale of the derived map is known (for example by user activity with the map), but the importance value is unknown. In this approach, using the radical law the number of features of the derived map can be calculated. But again there would be need to transform from number of features to importance values.

So there is need to investigate for transformation relation between importance value and number of features. The relation between these two quantities can be build based on some sample values and interpolating between them. Figure 3.7 shows two diagrams of experimental mapping between importance values and number of features. It should be mentioned that the mapping between these two quantities will highly be based on selection of importance value measure.

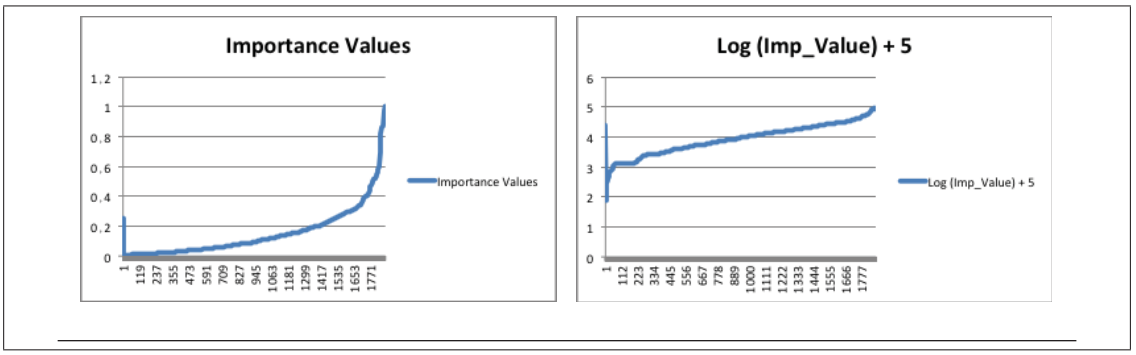

FIGURE 3.7: Two diagrams are shown. The left one shows the ordered importance values of a test dataset. As it can be seen the data values shows exponential growth. An effort of normalizing the values is shown on the right. The factor of 5 have just been added to result in positive numbers. The result is closer to a linear relation which can be appropriate for continuous zooming visualization to potential users.

3.4.2 Iterating Over The Forest

It was mentioned before in section 3.3.3, the real implementation of the *forest* potentially is being saved in a RDBMS but as in here the theoretical concepts will be discussed and the implementation details will be discussed in the next chapter.
As it was discussed before, the *forest* contains a set of trees who represent a partial network with a parent-child relation. The forest needs to be queried and individual trees are not connected to each other, so the data structure should be implemented with the possibility to list the trees. At query time the *forest* contains a set of trees that enables an iterator to go through trees individually. By visiting any tree, there would be possibility of traversing through it from the root to the leaves. Visiting any node of the tree, if the query parameter is between the minimum and maximum values of the node, that node would be added to the result set; if not the cursor will be passed to the child nodes of that individual node. If the node does not have any children, the iterator will go to the next tree in the *forest* set of trees.

3.4.3 Planar Slices

Considering the 3D visual corresponding formation of road network SSC, a planar slice is equivalent to search for the features whose importance range covers the query parameter. The process is consisted of fetching the set of appropriate features and visualizing them neatly. Appropriate features are fetched by iterating over the *forest* (this process includes traversing through individual trees). After fetching the segments, there would be need to apply smoothing. Smoothing should be done individually on any segment, therefore an iterator goes through the features. The line smoothing on any individual feature is done by dividing the feature range (difference of importance between maximum and minimum) to the number of feature nodes (except the first and the last nodes)

or mathematically:

$$\frac{(maximum - minimum)}{(N-2)} \tag{3.2}$$

where N is the total number of nodes in that edge. This is the number of nodes who should be eliminated as mentioned earlier in section 3.3.4.2. Having the levels and the query parameter, it should be clear how many of the nodes are being shown normally and which node should be displaced (and the displacement vector could be calculated). After this process the nodes and their positions for every segment are calculated. The nodes are connected with a polyline and the whole visualization is ready to be shown to the user.

3.4.4 Non-Planar Slices

Theory and implementation of non-planar slices of road network SSC needs more efforts to solve several questions. As it has been discussed in section 3.3.4, the 3D smoothing line simplification is an important requirement to perform such slices. Even if such slices are possible to perform, the result is not geometrically simpler. A potential solution in this field is using line morphing as suggested by Nöllenburg et al. (2008) but the investigation should consider this concept in 3D. Another aspect is the query parameter. In a planar slice, the parameter is constant in the whole dataset while in a non-planar slice, the parameter is a function of coordination.

Chapter 4

Implementation of Road Network SSC

4.1 General Information Regarding The Implementation

In the following sections, information about details of the software environment of the road network SSC is being discussed.

4.1.1 Programming Language

In order to implement the theoretical concepts and considerations, Python programming language[1] has been selected. Python is an interpreted language with potential of code implementation in several different programming paradigms including object oriented and procedural programming. This language is also considered as an open source language which give more freedom to potential programmers. The idea of a language being free also accelerates development of free softwares and libraries based on that language. Another important aspect of Python in the field of cartography, GIS and geomatics is the trend of using Python as built-in scripting languages inside several popular GIS softwares in the market. Some examples are ArcPy[2] scripting in ArcGIS[3], PyQGIS[4] scripting in Quantum GIS[5] and another implementation in GRASS GIS[6] [7].

[1] http://www.python.org Accessed on 5th of October 2013

[2] http://help.arcgis.com/en/arcgisdesktop/10.0/help/index.html Accessed on 5th of October 2013

[3] http://www.esri.com/software/arcgis Accessed on 5th of October 2013

[4] http://www.qgis.org/en/docs/pyqgis_developer_cookbook/intro.html Accessed on 5th of October 2013

[5] http://www.qgis.org Accessed on 5th of October 2013

[6] http://grass.osgeo.org/ Accessed on 5th of October 2013

[7] http://grasswiki.osgeo.org/wiki/Python Accessed on 5th of October 2013

In most cases the built-in scripting environment enables the users to automate data manipulation and geographical processing operations. There is also potential possibility of implementing Graphical User Interface (GUI) for user interactions with geoinformation.

4.1.2 RDBMS

The implementation of the road network SSC should have a data storage. The storage is an RDBMS while such systems give efficient means to store and fetch relational data. The RDBMS should also have spatial capabilities in order to efficiently work with geoinformation and also provide several spatial functionalities. PostgreSQL[8] is one of the popular open source RDBMSs. PostGIS[9] is an extension of PostgreSQL which enables it to manage geoinformation and also provides wide range of geometric and geographic functionalities. PostGIS has several characteristics which makes it an appropriate solution for geoinformation data handling:

- Can handle different geometric entities compatible with OpenGIS Simple Feature definition (OpenGIS Implementation Specification for geographic information - Simple feature access) [52]
- Supports WKT (Well-Known Text) and WKB (Well-Known Binary) feature definitions (OpenGIS Implementation Specification for geographic information - Simple feature access). There is also support for extended SRID-aware EWKT and EWKB formats (which are WKT and WKB extensions who support projections).
- Support for different projections and coordinate systems.
- Implementations of spatial indexes.
- Several implemented spatial functionalities who support geometric and topologic queries.
- Support for conversion to/from different data formats such as JSON (JavaScript Object Notation) and SVG (Scalable Vector Graphics).

In order to access PostgreSQL (and consequently PostGIS) through Python, Psycopg[10] library has been used.

[8] `http://www.postgresql.org` Accessed on 5th of October 2013
[9] `http://postgis.net/` Accessed on 5th of October 2013
[10] `http://initd.org/psycopg` Accessed on 5th of October 2013

4.1.3 Geometry Library

In order to access geometric features in the initial dataset, there would be need for a library which retrieves vector geometries. OGR (Simple Features Library)[11] is a sister library of well known raster library GDAL (Geospatial Data Abstraction Library)[12]. This vector library is originally a C++ library enabling programmers to access geometries who comply with OpenGIS Simple Feature definition (OpenGIS Implementation Specification for geographic information - Simple feature access). There is a Python wrapper available which enables Python programmers to use OGR functionalities from Python code. Several advantages of OGR are:

- Seamless integration with GDAL.
- Support for different geometry types compliant with OpenGIS Simple Feature definition.
- Data access in a layer-based formation.
- Support for different projections and coordinate systems.
- Data format conversion between common formats such as WKT, WKB, KML, GML and etc.

4.1.4 Graph Library

There would be also need for a library to create, query and analyze graph datasets. A promising library for Python is NetworkX [13]. It can be used for modeling simple, directed and multi graphs. Several network analysis algorithms have been already implemented in the default functionalities (such as common connectivity, centrality, clustering, flow analysis and etc.). The library is flexible about node and edges data and can handle data added to them. This capability can be used to add additional information of the nodes (like name, type and etc.) and edges (name, class, maximum speed and etc.). There is also flexibility about importing and exporting the graph being modeled. Beside querying nodes and edges of the road network graph, in the implementation mostly the connectivity evaluation has also been used.

[11] http://www.gdal.org/ogr/ Accessed on 5th of October 2013
[12] http://www.gdal.org Accessed on 5th of October 2013
[13] http://networkx.github.io Accessed on 5th of October 2013

4.2 Data Structure

Data entities being modeled in PostgreSQL database are *nodes* and *edges*. Both of them are modeled in separate tables in the database. Theoretically *nodes* and *edges* should be connected to each other by two means. One connection should define the inclusion relationship between an *edge* and its *nodes*. The other connection should define the sequence of *node* elimination based on the Douglas-Peucker line simplification algorithm. In the implementation these two connections has been combined in one table who covers both concepts.

Another concept which should be mentioned here is the edge forest structure of the road network SSC. As it has been mentioned in section 3.3.3, from an implementation point of view, the data storage (RDBMS) also holds the information of the forest. As the forest is set of trees (every individual tree may contain one or more tree-nodes), if the data structure saves a list (like a table in a database) of trees with possibility of iterating over them, it would be enough to represent the forest. Any individual tree itself is set of tree-nodes who may have parent-child relationship between them. The tree can be modeled by root-to-leaf and leaf-to-root models. Root-to-leaf models save the access information of the leaves in the root storage and leaf-to-root models save the access information of the leaf's parent (who can be a root or an inner node) in the leaf storage. A leaf-to-root approach has been used in the implementation, therefore any tree-node has a reference storage to reference its parent. If the reference storage is full, it points to the parent of that tree-node and if this reference storage is empty, that node is a root-node. The details of the data structure is given here and figure 4.1 shows the database diagram.

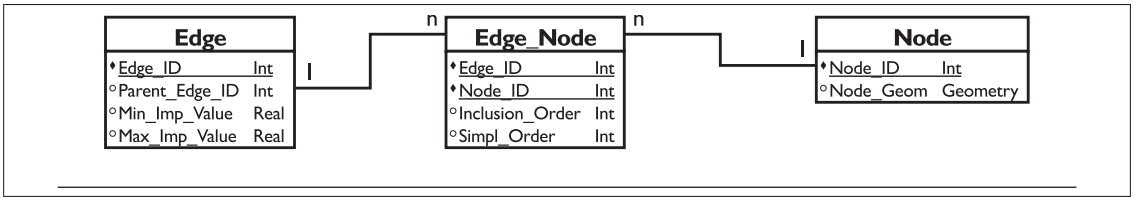

FIGURE 4.1: The diagram shows the tables and their relations. Nodes and Edges are being modeled in separate tables with their essential properties. The table in the middle illustrates the relation between the two entities.

4.2.1 Node

The nodes table defines the necessary information of individual nodes. To be able to distinguish nodes uniquely, a unique id property is necessary as a primary key. Besides an id, every nodes has geometry. Geometry is a data type implemented in PostGIS

to store geometry data. The data may be of point, line, polygon or other valid data type which complies with Open-GIS Simple Features definition. In this case the cell will contain point information of nodes (x and y values). There is possibility to add other information to nodes by extending the table.

4.2.2 Edge

The edges table defines the necessary information of individual edge beside reference information to model parent-child relations. To be able to distinguish edges uniquely, a unique id property is necessary as a primary key. Besides an id, every edge has two more columns who store the minimum and maximum importance values of the edge. In order to keep parent-child relationship, every edge has a cell containing its parent edge id. If the cell is empty, this edge will be a root and if not the id mentioned there point to the edge's parent, this is a foreign key to this table.

4.2.3 Edge-Node-Relation

As it was discussed before, two relations should be mentioned between nodes and edges. One connects two or more *nodes* to an *edge* and also defines the sequence of the nodes. Another connection defines the line simplification relationship. This is the sequence of *nodes* of the *edge* in which the order of the nodes are driven from Douglas-Peucker line simplification algorithm from the most important node to the least important node. The inclusion relationship is a *Many-to-Many* relation, meaning that an edge may contain more than one node and also one node may be in 1 or more edges (in case of junctions). That is the reason of forming the relation as a separate table. It should be considered that the case of a node being connected to more than 1 edge, happens only to the nodes who are in the beginning and end of edges but not the nodes in the middle. The simplification relationship is a *Many-to-One* relation, meaning that one edge contains many nodes (and not vice versa). It should be considered that the simplification relation only covers the nodes in the middle of the edge. The start and end nodes (points) will not go through the simplification process because if an edge only contains start and end nodes, it would be simplified only with the fade out (incremental transparency) operator. By considering both of the concepts, one may argue that there would be the possibility to have two separate tables of which one would portray inclusion and the other would portray simplification. The composition of these two relations together has one advantage which is simplifying the database query at fetch stage of SSC.

4.3 Software Architecture

The implemented software consists of several building blocks. As two main activities of the system are building and querying the SSC, more discussion will be focused on them.

4.3.1 More Detail on Building The SSC

The formation of the SSC is initiated by the Python module who reads the initial data. After the initial data processing stage, all of the needed information has been fetched and saved in the database and graph, so there would be no need to access the initial data again. Figure 4.2 illustrates the program setup at formation stage.

4.3.1.1 Initial Data Processing

In the implemented solution, the initial data is of ESRI[14] Shapefile [53] type. After accessing the Shapefile data through usage of OGR library, the road network is being read and added to the database and graph. The following pseudo-code demonstrates the process:

```
road_network_layer = read_from_shapefile("shapefile.shp")
g = graph()
db = postgres.connect()
foreach feature in road_network_layer.features:
{
     first_point = feature.firstpoint()
     last_point = feature.lastpoint()
     g.add_node(first_point)
     g.add_node(last_point)
     g.add_edge(first_point, last_point)
     edge_id = db.add_edge(feature.id)
     feature_dp_order = apply_douglas_peucker_ordered_sequence()
     points = feature.points
     foreach point in points:
     {
          point_id = db.add_node(point)
          loop_counter++
          point_dp_order = feature_dp_order[point]
```

[14] http://www.esri.com/ Accessed on 7th of October 2013

```
            db.add_edge_node_relation(edge_id, point_id, loop_counter, point_dp_order)
      }
}
```

One affective aspect is the importance values. The initial importance values should be calculated and added at this point. But as it was discussed before in the last chapter, the values would go through some changes during the graph weeding process. As segment length has been mentioned before to be an appropriate candidate for important values, the edge addition statement can be:

```
edge_id = db.add_edge(feature.id, feature.calculate_length)
```

In the case of edge betweenness centrality and based on the already implemented functionality in the NetworkX library, after building the whole graph the importance values can be calculated and updated in the database:

```
edge_importance_values = g.edge_betweenness_centrality()
foreach edge in db.get_all_edges():
{
      db.update_edge(edge.id, edge_importance_values[edge])
}
```

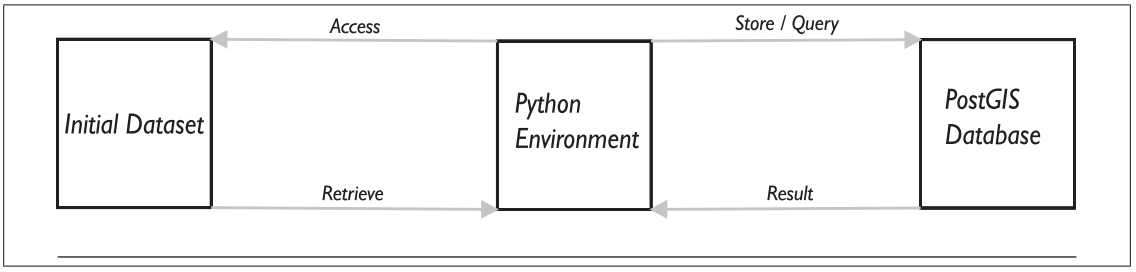

FIGURE 4.2: The main building blocks of the software environment is being shown at stage of formation of the road network SSC. The Python environment in the middle accesses the initial data and retrieves the needed data. While processing the data inside Python block, the data is being saved in the PostgreSQL data storage. Meanwhile there would probable necessity to query the data already saved in the PostgreSQL data storage.

4.3.1.2 Network Processing

After having the database and graph ready, graph weeding can take place. The theoretical background has been discussed in section 3.3.3 while keeping the the whole network connected the process decreases the nodes and edges of the graph and in some cases parent-child relations are derived from the network structure. From theory point of

view, there would be necessary to have a set of trees but from implementation point of view, such structure has been covered in the database design. More details can be found in sections 4.2 and 3.3.3. The following pseudo-code demonstrates the process:

```
queue_array = sort(db.fetch_all_imp_values())
foreach edge in queue_array[edges]:
{
     g_copy = g.copy()
     g_copy.remove_edge(edge)
     if (g_copy.isconnected):
     {
          g_copy.delete_edge(edge)
          queue_array.remove_first()
          foreach node in edge.nodes:
          {
               switch(node.degree):
                    0: g_copy.delete_node(node)
                    1: pass
                    2:
                    {
                         first_node = node.first_edge.first_node
                         last_node = node.last_edge.last_node
                         new_edge_id = g_copy.add_edge(first_node, last_node)
                         db.add_edge(new_edge_id)
                         foreach edge_of_node in node.edges:
                         {
                              queue_array.remove(edge_of_node)
                              g_copy.remove_edge(edge_of_node)
                              db.update_edge_parent(edge_of_node, new_edge_id)
                         }
                         queue_array.add(new_edge_id)
                         g_copy.delete_node(node)
                    }
                    default: pass
          g = g_copy.copy()
          }
     }
     else
     {
          queue_array.insert_at_end(edge)
          queue_array.remove_first()
     }
}
```

4.3.2 More Detail on Querying The SSC

Having the database with appropriate content (edges, nodes, relations, importance values), the system is ready to be queried. The query module should query the database for a scale or importance value. Discussion about mapping from one to the other has been already covered in section 3.4.1. By assuming that there is an importance value parameter available, the system functionality is being discussed here. The query stage is being illustrated in figure 4.3.

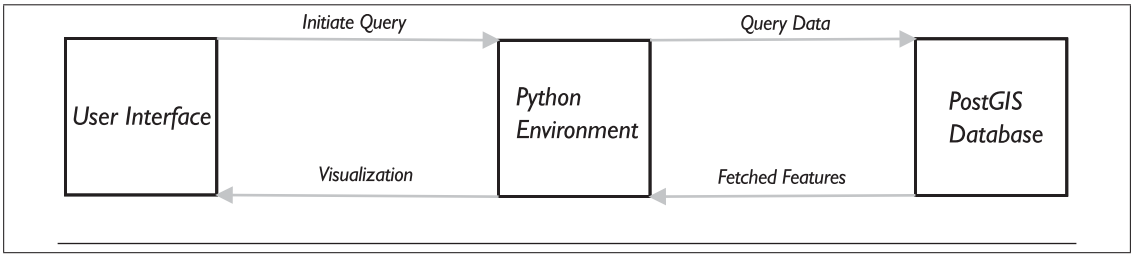

FIGURE 4.3: The main building blocks of the software environment is being shown at stage of querying the road network SSC. The process is initiated by user activity (e.g. zooming) therefore the user interface would query Python for new scale (importance value). The query would be passed to the PostgreSQL database and the features who fit with the query criteria will be sent back to Python and then to the interface.

4.3.2.1 Database Query

If the storage of the data is a forest, the query would start from the roots and continue to lower levels and check if the query parameter fits between the maximum and minimum value of that tree-node. In case of having a table, the edges are saved in a set of edges (keeping the parent-child relations through setting appropriate Edge_ID and Parent_Edge_ID). The following code queries the database and fetches valid features:

```
select Edge_ID from Edge
where (Min_Imp_Value < Query_Param) and (Max_Imp_Value >= Query_Param)
```

4.3.2.2 Building Geometry

As it has been discussed in section 3.3.1, the only entities who carry geometry are the nodes. The geometry is stored in a PostGIS-friendly format in the database. In order to build up the whole geometry, geometry of individual edges should be built and added to a set of results. In order to build the geometry of an edge, the appropriate nodes who are valid in the LoD corresponding with the query parameter should be fetched. Such

process is done first by querying the database for the appropriate features (mentioned in the last section). After having the edge features, the inter-LoD simplification takes place. Iterating over each edge, some nodes survive, some are being eliminated and some are being displaced. After fetching appropriate nodes, a line geometry should be composed out of the nodes. A function name ST_MakeLike[15] has been already implemented in PostGIS functions who does this task by having an array of points. So the result of the query can be passed to this function and it builds the geometry of the every edge.

4.3.2.3 Interface and Visualization

In order to have an interactive and easily accessible user interface, web browser has been selected as the interface where there is possibility of interacting with the data structure. The preview interface is a webpage who calls the Python query module. The module would call the database and the fetched geometry will be responded to the Python module. After this step the features should be visualized to the user. SVG format has been selected as the representation means because of three points: being able to visualize vector information, web browser friendly and also because it has promising future in the field of cartography and geovisualization. Since the features are of polyline or lingering type, they will be mapped to SVG *path* element[16]. Fortunately there is possibility to process partial SVG generation in PostgreSQL using PostGIS functionalities (ST_AsSVG[17]). This functionality will give the path element for every feature being sent to it as input parameter. The fetched features will be sent to this function and the result would be sent to Python query module. There would be some further processing (adding necessary headers for the document and optionally SVG styling). When the SVG is ready, it would be sent to the user to be shown on the web browser. A sample path from a road segment of Dresden (Germany) road network is given here (data from OpenStreetMap[18]):

```
<path d="M 13.7468588 51.0647456 L 13.7467282 51.0639843
13.7466772 51.0638025 13.746655 51.0637721" />
```

[15] http://postgis.refractions.net/documentation/manual-svn/ST_MakeLine.html Accessed on 7th of October 2013
[16] http://www.w3.org/TR/SVG/paths.html Accessed on 8th of October 2013
[17] http://postgis.net/docs/ST_AsSVG.html Accessed on 8th of October 2013
[18] http://postgis.net/docs/ST_AsSVG.html Accessed on 8th of October 2013

4.4 Results

Examples of results of the generalization process can be seen in figure 4.4. The top left illustration is the initial data being shown in Quantum GIS which contains 1941 features. Three queries have been sent to the dataset and the results are being shown. The first query result is showing 1716 features from a query with importance value of 0.00051. The second query result is showing 1184 features from a query with importance value of 0.04361. The third query result is showing 322 features from a query with importance value of 0.25663. It can be seen that generally the the network is shrinking from outside to inside to keep the more important features (since in urban areas, generally the center is more dense, when concerning road networks). Meanwhile in any region, the result is becoming less complex by weeding less important road segments.

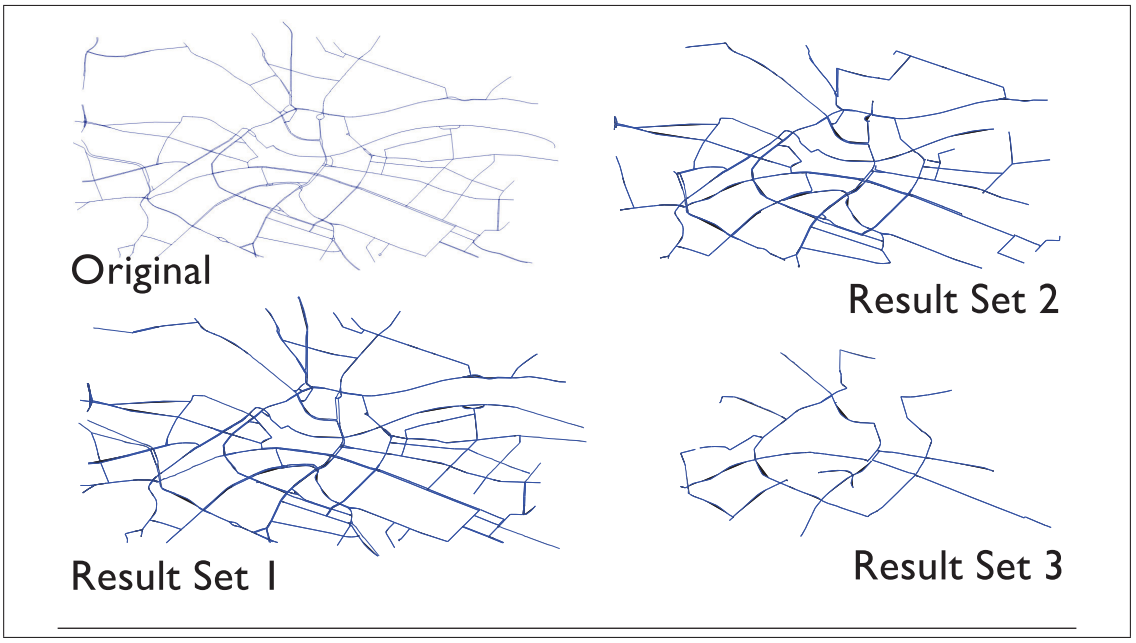

FIGURE 4.4: A sample dataset is being shown in this figure. Input data of the road network SSC is from major roads layer of Dresden, Germany. Simplifying generalization operator has not been active in this test.

Chapter 5

Conclusions and Outlook

Several conclusions can be derived from the outcomes of this research work. Firstly the adaptation of road networks datasets on SSC has been theoretically analyzed and also practically implemented. This research work is the first effort to solve and implement such adaptation. This achievement covered two main steps toward continuous zoomable (or vario-scale) road network maps:

- Building a hierarchical data structure on road networks. This solution is part of the road network SSC formation and can also be used in other generalization algorithms.
- Formulating the Pseudo-3D inter-LoD line simplifications based on Douglas-Peucker order of nodes. This process would also help in implementation of smoothing of line simplification in similar efforts in this field.

These two steps are generalizable to other future potential efforts who would rely on reactive data structures such as tGAP. There should be a strategy to build hierarchy and also a process of smoothing the generalization operator(s) of demand. More generally it can be concluded that there is room for more investigation of algorithms who enable GISs to provide more scale-flexible or scale-free representation of geoinformation. By these means beside solving critical research questions in the scientific research field, potentially the user experience of utilization of cartographic and geographic systems will change dramatically.
As this research work was the first effort to adapt road networks datasets on SSC, naturally there are open and unsolved question who will be research feed for future researchers. Some questions for potential future researches and investigations:

- Is it possible to form a complete framework of smooth generalization by combining algorithms of different geometry/geography data types?
- How can different weighting strategies of edges affect road network SSC results?
- How can deterministic relations between importance values and number of features be investigated and formalized?
- What strategy can be used to build a hierarchical forest on points datasets?
- Besides simplification, elimination and merge; how can other generalization operators be adapted to perform smoothly?
- How can non-planar slices be formulated and derived?

Bibliography

[1] AGENT, (1998). *Constraint Analysis. Deliverable A2.*

[2] Beard, M. K. (1991). *"Constraints on Rule Formation."* In Buttenfield, B. P., and R. B. McMaster (eds), Map Generalization: Making Rules for Knowledge Representation, Longman Group, pp. 121-135.

[3] Beck, Anatole, Michael N. Bleicher, and Donald Warren Crowe. (2000). *Excursions Into Mathematics: Millennium Edition.* Universities Press.

[4] Brandes, U. (2001) *"A faster algorithm for betweenness centrality."* Journal of Mathematical Sociology, vol. 25, no. 2, pp. 163-177.

[5] Brassel, Kurt E., and Robert Weibel. (1988). *"A review and conceptual framework of automated map generalization."* International Journal of Geographical Information System, vol. 2, no. 3, pp. 229-244.

[6] Burghardt, D., R. S. Purves, and A. J. Edwardes. (2004). *"Techniques for on the-fly generalisation of thematic point data using hierarchical data structures."* In Proceedings of the GIS Research UK 12th Annual Conference, pp. 28-30.

[7] Burghardt, Dirk, Stefan Schmid, and Jantien Stoter. (2007) *"Investigations on cartographic constraint formalisation."* In Workshop of the ICA Commission on Generalisation and Multiple Representation, vol. 19,p. 2.

[8] Cecconi, Alesandro, and Martin Galanda. (2002) *"Adaptive zooming in web cartography."* In Computer Graphics Forum, vol. 21, no. 4, pp. 787-799. Blackwell Publishing, Inc.

[9] Cecconi, A. (2003) *"Integration of Cartographic Generalisation and Multi-Scale Databases for Multi-Scale Databases for Enhanced Web Mapping."* PhD Dissertation, Ph. D. Thesis, Geographic Information System Division, Department of Geography, University of Zurich, Zurich, Switzerland.

[10] Douglas, David H., and Thomas K. Peucker. (1973) *"Algorithms for the reduction of the number of points required to represent a digitized line or its caricature."*

Cartographica: The International Journal for Geographic Information and Geovisualization, vol. 10, no. 2, pp. 112-122.

[11] Foerster, Theodor, Jantien Stoter, and B. Kobben. (2007). *"Towards a formal classification of generalization operators."* In Proceedings of the 23rd International Cartographic Conference.

[12] Galanda, Martin. (2003) *"Automated polygon generalization in a multi agent system."* Unpublished Ph. D. Dissertation, Department of Geography, University of Zurich, Switzerland.

[13] Grünreich D. (1985) *Computer-Assisted Generalisation* In: Papers CERCO Cartography Course, Institut Für Angewandte Geodäsie, Frankfurt am Main, Germany.

[14] Guttman, Antonin. (1984) *"R-trees: A dynamic index structure for spatial searching."* Vol. 14, no. 2. ACM.

[15] Haggett, Peter. (1967) *"Network models in geography."* Models in geography , pp. 609-668.

[16] Horton, Robert E. (1945). *"Erosional development of streams and their drainage basins; hydrophysical approach to quantitative morphology."* Geological Society of America Bulletin, vol. 56, no. 3, pp. 275-370.

[17] International Cartographic Association. (1973). Commission Definition, Classification and Standardization of Technical Terms in Cartography. *"Multilingual dictionary of technical terms in cartography."* Steiner.

[18] Jiang, Bin, and Christophe Claramunt. (2004). *"A structural approach to the model generalization of an urban street network."* GeoInformatica vol. 8, no. 2, pp. 157-171. 2004.

[19] Keates John S. (1989) *"Cartographic Design and Production"* 2nd Ed, Longman, London.

[20] Li, Zhilin, and Stan Openshaw. (1992) *"Algorithms for automated line generalization based on a natural principle of objective generalization."* International Journal of Geographical Information Systems, vol. 6, no. 5, pp. 373-389.

[21] Longley, Paul, ed. (2005). *"Geographic information systems and science"* John Wiley & Sons.

[22] Mackaness, William A., and Kate M. Beard. (1993). *"Use of graph theory to support map generalization."* Cartography and Geographic Information Systems, vol. 20, no. 4, pp.210-221.

[23] McMaster, Robert Brainerd, and K. Stuart Shea. (1992). *"Generalization in digital cartography."* Washington, DC: Association of American Geographers,

[24] Meijers, Martijn, and Peter van Oosterom. (2011) *"The space-scale cube: An integrated model for 2D polygonal areas and scale."* In 28th Urban Data Management Symposium, vol. 38, pp. 95-102.

[25] Meijers, Martijn. (2006). *"Implementation and testing of variable scale topological data structures-Experiences with the GAP-face tree and GAP-edge forest."*

[26] Müller, J. C. (1991). *"Generalization of spatial databases."* Geographic Information Systems (eds Maguire, DJ, Goodchild, MF and Rhind, DW) 1, pp. 457-475.

[27] Müller, J. Jean-Claude, J. Jean-Philippe Lagrange, and Robert Weibel, eds. (1995). *"GIS [ie Geographic Information Systems] and Generalization: Methodology and Practice."* No. 1, CRC Press.

[28] Nickerson, Bradford G. (1988). *"Automated cartographic generalization for linear features."* Cartographica: The International Journal for Geographic Information and Geovisualization, vol. 25, no. 3, pp. 15-66.

[29] Nöllenburg, Martin, Damian Merrick, Alexander Wolff, and Marc Benkert. (2008). *"Morphing polylines: A step towards continuous generalization."* Computers, Environment and Urban Systems, vol. 32, no. 4, pp. 248-260.

[30] O'Sullivan, David, and David John Unwin. (2003). *"Geographic information analysis."* John Wiley & Sons.

[31] Peter, Beat, and Robert Weibel. (1999) *"Using vector and raster-based techniques in categorical map generalization."* In Third ICA workshop on progress in automated map generalization, pp. 12-14, Ottawa, Canada.

[32] Raisz, Erwin. (1962). *"Principles of Cartography."* New York: McGraw-Hill.

[33] Robinson, Arthur, Randall Sale, and Joel Morrison. (1978). *"Elements of Cartography."* 4th Edition, New York: John Wiley & Sons, 2003.

[34] Ruas, Anne, and Corinne Plazanet. (1996). *"Strategies for automated generalization."* In Proceedings of 7th International Symposium on Spatial Data Handling, vol. 1, no. 6, Delft: Delft University of Technology.

[35] Sester, Monika, and Claus Brenner. (2005). *"Continuous generalization for visualization on small mobile devices."* In Developments in Spatial Data Handling, pp. 355-368, Springer Berlin Heidelberg.

[36] Steiniger, Stefan, and Robert Weibel. (2005). *"A conceptual framework for automated generalization and its application to geologic and soil maps."* In Proceedings of XXII Int. Cartographic Conference.

[37] Strahler A. N. (1960) *Physical Geography* Wiley, New York.

[38] Thomson, Robert C., and Dianne E. Richardson. (1995). *"A graph theory approach to road network generalisation."* In Proceeding of the 17th International Cartographic Conference, pp. 1871-1880.

[39] Thomson, Robert C., and Dianne E. Richardson. (1999). *"The 'good continuation' principle of perceptual organization applied to the generalization of road networks."*.

[40] Töpfer, Friedrich, and W. Pillewizer. (1966). *"The principles of selection."* Cartographic Journal, The Vol. 3, no. 1, pp. 10-16.

[41] Van Kreveld, Marc. (2001). *"Smooth generalization for continuous zooming."* In Proc. 20th Intl. Geographic Conference, pp. 2180-2185.

[42] Van Oosterom, Peter. (1991) *"The reactive-tree: A storage structure for a seamless, scaleless geographic database."* Auto-Carto 10: Technical Papers of the 1991 ACSM-ASPRS Annual Convention, Baltimore: ACSM-ASPRS, vol. 6, pp. 393-407.

[43] Van Oosterom, Peter. (1995). *"The GAP-tree, an approach to ?on-the-fly?map generalization of an area partitioning."* GIS and Generalization: Methodology and Practise, Taylor & Francis, London, pp. 120-132.

[44] Van Oosterom, Peter. (2005). *"Variable-scale topological data structures suitable for progressive data transfer: The GAP-face tree and GAP-edge forest."* Cartography and Geographic Information Science, vol. 32, no. 4, pp. 331-346.

[45] Van Oosterom, Peter. (2009). *"Research and development in geo-information generalisation and multiple representation."* Computers, Environment and Urban Systems, vol. 33, no. 5, pp. 303-310.

[46] Van Oosterom, Peter, and Martijn Meijers. (2011). *"Towards a true vario-scale structure supporting smooth-zoom."* In Proceedings of 14th ICA/ISPRS workshop on generalisation and multiple representation, Paris, pp. 1-19.

[47] Visvalingam, Mahes, and Peter J. Williamson. (1995). *"Simplification and generalization of large scale data for roads: a comparison of two filtering algorithms."* Cartography and Geographic Information Systems, vol. 22, no. 4, pp. 264-275.

[48] Weibel, Robert. (1997). *"Generalization of spatial data: Principles and selected algorithms."* In Algorithmic foundations of geographic information systems, pp. 99-152. Springer Berlin Heidelberg.

[49] Weibel, Robert, and Geoffrey Dutton. (1998). *"Constraint-based automated map generalization."* In Proceedings of the 8th International Symposium on Spatial Data Handling, pp. 214-224, Columbia (British): Taylor & Francis.

[50] Weibel, Robert, and Geoffrey Dutton. (1999). *"Generalising spatial data and dealing with multiple representations."* Geographical information systems, vol. 1, pp. 125-155.

[51] http://primes.utm.edu/cgi-bin/caldwell/tutor/graph/glossary.html Accessed on 6th of October 2013

[52] http://portal.opengeospatial.org/files/?artifact_id=25355 Accessed on 2nd of October 2013

[53] http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf Accessed on 7th of October 2013