Department of Geomatics Engineering

A New Approach for Simplification of Linear Vector Data for Internet_based GIS Applications

(URL: http://www.geomatics.ucalgary.ca/links/GradTheses.html)

by

Nadia Shahriari Namini

November 2002

UNIVERSITY OF
CALGARY

THE UNIVERSITY OF CALGARY

A NEW APPROACH FOR SIMPLIFICATION OF LINEAR VECTOR DATA

FOR INTERNET_BASED GIS APPLICATIONS

by

NADIA SHAHRIARI NAMINI

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE

DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF GEOMATICS ENGINEERING

CALGARY, ALBERTA

NOVEMBER, 2002

ABSTRACT

Over the past two decades, Geographic Information Systems (GISs) have been widely used in many applications. Nowadays, many Internet-based GISs provide the users with the most up-to-date geodata and on-line geoprocessing services. One of the main problems with Internet-based GIS technology is that the speed of the data transferring is affected by the speed of the Internet connection and the type of the data to be transferred. Data transmitting speed is a very important issue in Internet-based GISs. Although improving networking technology and speed can reduce the cost of electronically transmitting of geospatial data through Internet, however, this issue still remains a challenge.

To optimize Internet-based GIS applications, line simplification for vector data compression has been investigated in this thesis. Different data sets have been used in the analysis. The results show that line simplification significantly reduces the size of vector geospatial data sets, however, it introduces positional errors in the data. A new approach, using adaptive tolerance value has been introduced to minimize positional errors. The results clearly show that positional accuracy of data is maintained to the user specified level of accuracy. Segmentation of lines to more homogenous segments resulted in better simplification from data quality point of view. Code compression was integrated with line simplification to maximize the compression ratio. The results showed that the effect of code compression on vector data is insignificant and time consuming.

ACKNOWLEDGEMENTS

This thesis is dedicated to

Ali, my dear husband and partner in life

And

Our two lovely sons, Aidin and Armin

Without their supports this could never have happened.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

DCT: Discrete Cosine Transform

DSAD: Digitally Sampled Analog Data

ESRI: Environmental Systems Research Institute

GIF: Graphic Interchange File

GIS: Geographic Information System

HTTP: Hyper Text Transfer Protocol

JPEG: Joint Photographic Experts Group

MPEG: Moving Pictures Experts Group

NTDB: National Topographic DataBase

PC: Personal Computer

URL: Uniform Resource Locator

UTM: Universal Transverse Mercator

Chapter 1

INTRODUCTION

1.1   Background

Over the past two decades, Geographic Information Systems (GISs) have been widely used in support of urban and regional planning, agriculture, forestry, facilities management, environmental studies and many other fields. Together with the use of the Internet, GIS could be further developed to allow many more people to have access to geospatial data and geoprocessing services. The Internet has already become a huge source of geospatial information and offers different GIS functionalities. Nowadays, many Internet-based GISs provide the users with the most up-to-date geodata and on-line geoprocessing services. Using Internet-based GIS, users are able to access GIS applications by only using a web browser without purchasing or installing any GIS software in their PCs. Due to the many advantages of Internet-based GIS technology, the number of users of this technology is rapidly increasing and therefore, the demands for access to geospatial data and geoprocessing services through the Internet should also continue to increase.

Internet-based GISs are based on client/server architecture. Weise (2001) defines this architecture as follows.

Client and server, these are two programs communicating across a computer network either on Internet (global public collection of computer networks) or on Intranet (private network based on Internet standards) by means of a HTTP (Hyper Text Transfer Protocol) that is a kind of communication language. The most common client program is a Web browser (eg. Netscape Navigator, Microsoft Internet Explorer). By means of a URL the browser program sends a request to the server and finally a file is transmitted from the server to the client. The server side usually consists of many technologies that are working together such as...

Internet-based GISs may be implemented using different strategies. Based on the workload, heavy/light server and thick/thin client are used in different Internet-based GISs. Server-side strategies use heavy server and thin client. In these strategies, data/GIS software is centralized in server side and client uses a Web browser to communicate with the server. A single centralized data/software site is generally cheaper, easier to keep updated, and uses computer power more efficiently than distributing everything (Plewe, 1997). Client-side strategies, on the other hand, use light server and thick client. In these strategies most processing is handled by the client. In these strategies users at the client side download small programs called applets to add some GIS capabilities to their browser software. Usually these applets enable users to perform very basic GIS functions such as panning and zooming. Both strategies have advantages and disadvantages. Plewe (1997) mentions some of these advantages and disadvantages as follows:

A thick client (interacting with a light server) allows for flexible and powerful analysis, but cannot handle as much network traffic, limits your audience, and can be a nightmare to maintain (i.e. keeping all of the clients up to date). A thin client/heavy server setup uses a considerable amount of bandwidth sending so many maps, and is generally limited to more simple applications. However, it can be used by many more people, including those not willing to download and install a thick client just to use services.

To overcome the disadvantages of client and server strategies, a hybrid approach can be used.

One of the main problems with Internet-based GIS technology (especially with thin client and hybrid approaches) is that the speed of the data transferring is affected by the speed of the Internet connection and the type of the data to be transferred (Figure 1.1). Data transmitting speed is a very important issue in Internet-based GISs. Although improving networking technology and speed can reduce the cost of electronically transmitting of geospatial data through Internet, however, this issue still remains a challenge.
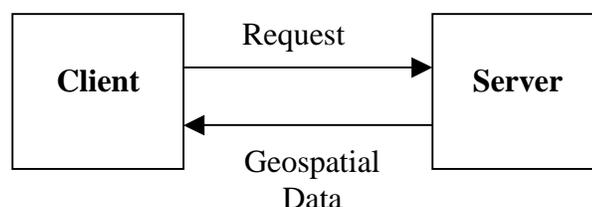
Figure 1.1 Geospatial Data Transmission through Internet

Geospatial data is usually categorized as raster or vector data. Because these two data models have completely different data structures, Internet transmissions of raster and vector data requires different handling issues. Efficient transmission of raster data through Internet has already been very successful due to the nature of raster data model:

> *The raster model proves efficient for a number of reasons. File size remains predictable for a given resolution, and constant regardless of feature complexity. Moreover, transmission of raster data can proceed incrementally by means of a simple algorithm. An initial (very coarse resolution) version is compromised of a subset of the raster file's rows and columns. Detail is added by transmitting additional rows and columns, in a randomized order. At the viewer's workstation, the Internet browser displays transmitted data with the visual effect of a blurred image whose details gradually sharpen. This method works well for digital images, orthophotography, and satellite data products. (Buttenfield, 1999)*

In addition, many data compression algorithms have already been developed to reduce the file size of raster data. However, transmission of vector geospatial data through Internet still remains a challenge. Vector data model has completely different structure than raster data model in the sense that it's not comprised of rows and columns. Many of the GIS analysis functions use vector data. Vector files are usually large and the file size increases with the increase of the geometric complexity of features in the file. Common code compression algorithms do not significantly reduce the size of vector data files. However, other way can be used to reduce the size of vector data file through the use of simplification algorithms. McMaster (1989) states simplification routines are applied for data compaction purposes. These algorithms remove redundant or unnecessary coordinate pairs from the vector data based on some geometric criterion, such as distance between points.

The Internet-based GIS users requesting vector data often require a less detailed version of the original data. User, in client side, requests the data, visualizes it upon receiving and then makes a query and/or request for spatial analysis. While the original data is kept in the server side, the simplified version of data can be sent to user. If identification numbers of geospatial features after transmission remain the same as the original data, any query, processing and analysis could be done using original data in server side without loosing any information due to the simplification. Due to the file size reduction, transmission of simplified vector data will definitely be faster than the original file. However, the important point is to let the user to select the level of simplification or quality of data regarding the positional displacements or the level of geometric complexity of features to be preserved after simplification. Therefore, new approaches need to be designed and implemented to

fulfill these requirements. The most important aspect of this research is mainly to develop and test a new approach that meets these requirements. To minimize the vector map file size, code compression and content compression algorithms may also be integrated.

1.2 Objectives

The main focus of this research work is to investigate line simplification as a vector geospatial data compression technique and to develop and test a new approach for line simplification of vector linear geospatial features in which the positional accuracy and changes in geometric characteristics of features are under user's control. The research focuses on geospatial linear features such as roads and rivers. The followings have been identified as the major objectives of this thesis:

- To implement a line simplification algorithm as a content compression technique to reduce the vector data file size.
- To investigate the effect of line simplification on the geometric complexity of linear features. This involves implementing some linear complexity measures.
- To design a new approach to optimize line simplification algorithm for linear features with different geometric complexities and also non-homogenous lines.
- To implement the new approach to preserve the positional displacements and/or the geometric complexity of linear features to the levels specified by the user based on the application.
- To integrate the code compression with content compression algorithms to minimize the file size.

It should be noted that the software, which will be developed to fulfill these objectives, doesn't need to have a window-based interface as it is supposed to be used by Internet-based GIS software.

1.3 Thesis Outline

The objectives of this research have been defined in the previous section. This thesis basically consists of seven chapters. The outline of each chapter has been described below.

Chapter 1 contains background information and the statement of the problem. The objectives of this research and thesis outline are also discussed in this chapter.

A general overview of data compression is given in Chapter 2. This chapter also introduces the bases of the two major categories of data compression; namely, lossy and lossless compressions. As the goal of this chapter is only providing a general overview about data compression techniques, none of the compression techniques will be discussed in details. A discussion about content compression for vector geospatial data is also given in this chapter.

An overview of line simplification theory and algorithms is given in Chapter 3. The main categories of line simplification algorithms and their main characteristics are discussed in this chapter. Douglas-Peucker line simplification algorithm is discussed in more details.

The results of Douglas-Peucker algorithm on some geospatial vector data sets are presented and evaluated in this chapter.

Chapter 4 contains the mathematical measures for line complexity. Four main categories of complexity measures have been addressed. The results of line complexity measures on a data set are presented and analyzed in this chapter.

Line simplification algorithms induce positional errors in data sets. The amount of these errors differs for lines with different geometric complexity. Using a constant tolerance value for simplification of all the lines in a data set induces different amounts of displacement errors for different lines. The displacement errors introduced by line simplification algorithm have been discussed in Chapter 5. One displacement measure is implemented and the result of this measurement on the data set is presented and evaluated. A new approach to minimize these positional displacements is proposed, implemented and applied on the data set and the result is presented and analyzed in this chapter. In this proposed approach lines are simplified using adaptive tolerance value rather than a constant tolerance value.

The linear features of the map are not necessarily homogeneous. Therefore using one tolerance value to simplify a non-homogenous line introduces very different amounts of positional errors in different parts of the line. Segmentation of lines to homogenous sections based on geometric complexity of the line is a solution for this problem. These issues are discussed in Chapter 6. The other important issue discussed in this chapter is

about the way to select the tolerance value for each segment of line in such a way that the user-specified level of geometric characteristics be preserved after simplification. This approach is implemented and the results of applying it on the data set is presented and evaluated in this chapter. Line simplification is basically a content compression algorithm. To minimize the vector map file size, code compression and content compression algorithms are integrated. This issue is also discussed in this chapter. This chapter also includes a discussion about decoding the file after compression. The result of integrated algorithms on data set is presented and analyzed in this chapter.

Finally, in Chapter 7 conclusions based on the research work will be outlined. A number of recommendations for future work are also included in this chapter.

Chapter 2

DATA COMPRESSION

2.1 Introduction

Data compression is a very important topic and is used in many applications. Compression relies on the fact that the data is redundant and data compression algorithm should remove this redundancy, leaving only the informational content. Therefore, in most cases the size of compressed file is smaller than that of the original file. Data compression is often referred to as data encoding, and decompression is often called data decoding. Data compression algorithm is sometimes called as compressor.

Data compression has very important application in data transmission and data storage. The aim of data compression is to minimize the amount of data to be transmitted or stored. Nowadays, the number of applications, which require storage of large volumes of data, is constantly increasing. Storage space on disks is expensive and a compressed file occupies less disk space than an uncompressed file. Therefore, data compression reduces the storage costs. On the other hand, the rapidly increase in computer communication networks is resulting in massive transfer of data over communication links. Data compression increases

the speed of data transfer because a smaller file can be transferred faster than a larger file. This results in increasing the capacity of the communication channel.

The most important factor to evaluate data compression algorithms is, mainly, the amount of compression they provide. This amount can be measured by a compression ratio, which has been defined in several ways. One way to calculate compression ratio is using percentage.

> *You can also measure using %, (compressed_length / original_length)*100, for example if we had a file with a size of 400 bytes and compress it down to 100 bytes, the ratio will be (100/400)*100 = 25% so the output compressed file is only 25% of the original. However there's other method: (1 - (compressed_length / original_length))*100. In this case the ratio is %75 meaning that we have subtracted 75% of the original file. In both cases the compression is the same, but the ratios are different, ...(Campos, 2000)*

Sometimes compression ratio is represented as a very simple ratio such as 2:1, meaning that the size of the compressed file is half of the size of the original file. This measure is usually used when the exact ratio of compression is not needed. The amount of compression ratio depends on the characteristics of the source data to great extent. Even given a specific type of file, the contents of a file, particularly the orderliness and redundancy of the data, can strongly influence the compression ratio (Goebel, 2002).

The speed of compression is also another important factor specially when compression is used for data transmission purposes.

> *When data compression is used in a data transmission application, the goal is speed. Speed of transmission depends upon the number of bits sent, the time required for the encoder to generate the coded message, and the time required for the decoder to recover the original ensemble. (Lelewer and Hirschberg, 1987)*

The speed of compression can be measured in terms of kilobytes per second. It is calculated by this formula: (file_size / compression_time).

This chapter introduces the bases of the two major categories of data compression; namely, lossy and lossless compressions. As the goal of this chapter is only providing a general overview about data compression techniques, none of the compression techniques will be discussed in details. A discussion about content-based compression for vector geospatial data is also given in this chapter.

2.2 Data Compression Forms

In terms of data recovery, data compression techniques are, in general, divided into two categories: lossy and lossless compressions. Lossy and lossless terms describe whether or not all original data can be recovered when the file is uncompressed. These two categories of data compression are described in the following sub-sections.

2.2.1   Lossy Compression

Lossy compression refers to data compression techniques in which some amount of data is lost after compression. Lossy compression techniques permanently remove redundant or unnecessary information from the file. With lossy compression, the original file can never be recovered exactly as it was before compression. It means decompressed file will not be the same as original file after lossy compression is applied to the file. Since some data are lost in lossy compression techniques, these techniques are usually applied to data where the lost data is not important. Sounds, images and videos are all signals. Digital signals are an

imperfect representation of an analogical signal, thus in compressing them we can discard some of the information to achieve more compression (Campos, 2000). However lossy compression should not be used for critical data.

> *…, it is not a good method of compression for critical data, such as textual data. It is most useful for Digitally Sampled Analog Data (DSAD). DSAD consists mostly of sound, video, graphics, or picture files. Algorithms for lossy compression of DSAD vary, but many use a threshold level truncation. This means that a level is chosen past which all data is truncated. In a sound file, for example, the very high and low frequencies, which the human ear cannot hear, may be truncated from the file. Some examples of lossy data compression are JPEG, MPEG, and Indeo. (Ladino, 1996)*

JPEG is an image compression technology and MPEG and Indeo are video compression technologies. Sometimes, lossy compression technologies let the user to decide how much loss to be introduced in the file. Users usually make a trade-off between file size and the quality of data after compression. In most of the cases signal compression goes of discarding as much data as possible but retaining as much quality as possible (Campos, 2000).

2.2.2   Lossless Compression

Lossless compression refers to data compression techniques in which no data is lost during the compression. With lossless compression, every single bit of data in the file is restored after the file is uncompressed.

> *Lossless data compression works by finding repeated patterns in a message and encoding those patterns in an efficient manner. For this reason, lossless data compression is also referred to as redundancy reduction. Because redundancy reduction is dependant on patterns in the message, it does not work well on random messages. Lossless data compression is ideal for text. (Ladino, 1996)*

Because no data should be lost in text, spreadsheets, databases and programs, lossless compression techniques should be used when compressing these types of data. Huffman coding is an example of lossless compression algorithm. The Huffman compression algorithm compresses files by assigning smaller codes to frequently used characters and longer codes for characters that are less frequently used (A code is a sequence of zeros and ones that can uniquely represent a character) (Saju, 2002).

Sounds, videos and images can also be lossless compressed, especially when storing their master sources, though then the compression achieved is far worse than with lossy compression. The Graphic Interchange File (GIF) provides lossless compression for images.

2.2.3   Combination of Lossy and Lossless Compressions

Lossy and lossless techniques are sometimes used together to obtain the highest compression ratios. This integrated approach has especially important application in some areas such as military and medicine, where some parts of the images are very important while the other parts do not contain very critical data. In this approach the significant parts of the image are losslessly compressed to preserve the local details while the other parts of the image are lossily compressed to remove the redundant information. Varga (1998) presented such an approach in which Quadtree structure has been used for lossless compression of areas of interest in an image and JPEG has been used for lossy compression of other parts of the image. This is in fact a content-based compression approach in which the contents of the image are first analyzed and categorized to important and non-important

parts and then lossless and lossy compression techniques are applied for these parts respectively.

The integrated approach can be used in any application in which losing some information is valid, however, the amount of the lost information and the way to specify those parts of data vary in different applications.

2.3 Compression of Geospatial Data

Geospatial data are stored either in vector or in raster format. Vector and raster data models have completely different structures to store the geospatial data. In vector data model, geospatial data is represented in the form of coordinates and the basic units of spatial information are points, lines and polygons. Each of these basic units is composed of one or more coordinate pairs. In raster data model, each area is divided into rows and columns, which form a regular grid structure. Each cell has a value, which is in fact the attribute value of that cell. The spatial location of each cell is implicitly contained within the ordering of the pixels. Geospatial data, either in raster or in vector format, need to be compressed for storage and transmitting purposes. Some compression methods can be applied to both vector and raster data files, while some cannot due to the different data structures of these two models.

In a simple-structure raster file, every pixel is given a single value. In this case, there is no compression because many like values are stored in the file. However, there are many raster data models, which are used to store raster data in more compressed form. Run-length

coding and Quadtree are some of these models. The compression achieved by these data models is lossless and these models work best when there are large areas of equal value pixels. Code compression techniques such as Huffman coding are lossless code compression methods and can be used to compress raster data files. The main shortcoming of lossless compression techniques is limited amount of compression. There are also lossy transform domain compression techniques in which the raster data is represented using an appropriate basic set. Discrete Cosine Transform (DCT) based compression and wavelet transform are two examples of transform domain methods (Vemuri, et. al., 2002). JPEG lossy compression technique is a DCT based compression method.

Geospatial vector data files (such as roads and rivers) are usually big files and need compression for storage and transmitting purposes. Code compression techniques such as Huffman coding can be used to compress vector data files. These compression methods are lossless and therefore, result in limited amount of compression. As it was mentioned before, the basic units of vector map data (namely linear features) are points, lines and polygons that are composed of one or more pairs of coordinates. Therefore, compression can be achieved by generalizing the features, i.e. removing unnecessary details from them. Of course, this type of compression will be lossy compression as it permanently removes some parts of data. Fourier and Wavelet transforms have been used to generalize vector geospatial data:

> The mainspring is that we want to represent a curve by a set of coefficients, each coefficient being related to a spatial frequency, hence with a wavelength that approximates the size of the features detected. The objective is to apply different operations according to the size of this wavelength: The low frequency coefficients characterize the global shapes and have to be

kept unchanged. The high frequency coefficients represent the small details and must be cut off. (Plazanet, et al., 1996)

Ioup (2000) has used wavelet transforms to compress vector map data. She states that by using only selected transform coefficients, and/or modifying them, desirable results such as noise removal or compression may be obtained. In her approach she has applied wavelet transforms on longitudes and latitudes separately. She has used different wavelets to compress two vector map data sets and compared the results. Fourier Transformation has also been used to remove the unnecessary details from the linear features. Research has shown some problems in using Fourier Transformation for generalization.

Some intermediate details show a strong resistance to the smoothing and a drift in the curve location can appear, which may induce topological defeats. Also, the bends are generally shortened by the enlargement... No matter what other filters are used to reduce these defeats, they persist, and are explained by lack of spatial information in the Fourier representation. (Plazanet, et al., 1996)

Wavelet transforms have been used in the same research work by Plazanet, et al. (1996). Though the results have been much better compared to using Fourier transformation, however, the problems of the location drifts and the shortening of the bends have not been solved.

Another way to compress vector map data is simplification of linear features. This is also a lossy compression approach, in which the unnecessary details are removed by deleting some points from the linear features. The potential advantage of this approach compared to transformation approach is that the simplification can be controlled by selecting a tolerance value based on the map positional accuracy, while in transformation methods the threshold is set by keeping some coefficients and setting the others equal to zero without any direct

relation to the positional map accuracy. The other potential advantage of simplification approach is that in this approach we can keep the most critical points in their original positions and only remove the non-critical points. Line simplification should be categorized as a content-based compression technique, because the contents (shape of linear features) are taken into consideration for compression rather than numbers indicating coordinates of points composing the linear features. Line simplification output can be compressed further using a lossless compression technique.

This research will investigate the application of line simplification algorithm in compressing linear vector map data and will focus on map positional accuracy issues during the lossy compression of the data.

2.4 Summary

Data Compression removes the redundant data from the file, leaving only informational content resulting in reduced file size. This is desirable for data storage and data transmission. There are lossy and lossless forms of data compression. Lossy data compression is usually used when data doesn't have to be restored perfectly. Lossless data compression is used when the data has to be uncompressed exactly as it was before compression. With lossless methods, the amount of compression is strictly limited. Lossy compression is usually used for images, sounds and videos, where the data is still of acceptable quality after losing some information during the compression. Lossless compression is used for critical data such as text, spreadsheets, databases and programs.

Geospatial data are stored either in vector or in raster format. Geospatial data need to be compressed for storage and transmitting purposes. Many compression techniques have already been developed to compress raster data. Lossless code compression techniques can be used to compress vector data as well. However, there are not much research works done for content-based compression of vector map data. Content-based compression techniques for vector geospatial data are techniques in which the contents (shape of linear features) are taken into consideration for compression. Lossy transform domain compression techniques such as Fourier and Wavelet transforms are being investigated. These techniques have some disadvantage. Line simplification approach can also be used to compress vector map data and has some potential advantages compared to transformation techniques, namely the potential possibility of selecting a tolerance value based on the map positional accuracy and keeping the most critical points in their original locations. Line simplification output can be compressed further using a lossless code compression technique.

Chapter 3

OVERVIEW OF LINE SIMPLIFICATION THEORY AND ALGORITHMS

3.1 Introduction

Cartographic generalization schematically consists of selecting the features to be maintained at the targeted scale, simplifying non-relevant characteristics, enhancing significant shapes, displacing without defacing global and local shapes and finally harmonizing the final aspect (Plazanet, 1997). Until about a decade ago, map generalization was practiced by trained cartographers, who learned it by using instructions, examples and intuition. Traditionally, map-making agencies were practicing manual cartographic generalization which involves a large set of generalization techniques. Each organization had its own guidelines, standards and procedures for manual map generalization in fulfillment of its mission. Only recently has this situation begun to change, but it has done so dramatically and definitively, as maps – along with every other form of human communication – have become digital (Dutton, 1999).

Recent reviews of digital generalization process have identified a number of distinct processes that are involved. Most of the techniques focus on either the manipulation of

vector- or raster-mode data. Much of the work on the generalization of raster-mode images has been mainly in the field of remote sensing and includes techniques such as high- and low-frequency filtering (McMaster, 1989). High-frequency filters emphasize fine details and edges by passing the higher frequencies while low-frequency filters emphasize the more generalized trends. Fourier analysis, two-dimensional convolution, linear edge detection, nonlinear edge enhancement are some of the common generalization techniques for raster imagery. Since the mid-1960s, however, most work in this area has addressed generalization in vector-mode. Specifically, researchers have addressed the problems of the representation of lines on maps in digital mode. Algorithms have been developed for the simplification, smoothing, enhancement, displacement and merging of linear features (McMaster, 1989).

Line simplification is an important function in cartography and GIS. In line simplification source data are transformed to reduce data volume, to merge databases with different scales, or to maintain cartographic quality when scale is reduced. Line simplification involves the selective elimination of vertices along a cartographic line to remove unwanted information. Line simplification is only one component of cartographic generalization. However, it is arguably the most commonly applied generalization operation and is widely used in commercial GIS software packages (Veregin and Dai, 1999).

3.2 Line Simplification Algorithms

Several line simplification algorithms have been presented by researchers in different disciplines such as cartography, computer science and mathematics. Most simplification

algorithms weed from unnecessary points based on some geometric criterion, such as distance between points or displacement from a centerline.

> *Most simplification algorithms incorporate some mechanism to control the amount of detail that is removed; for example, in an 'nth point' algorithm, the n refers to a numeric threshold (a tolerance value) determining that 1/n points will be eliminated systematically or randomly (Tobler, 1966). Tolerance values can take many forms. They provide the width of corridors within which coordinates are eliminated (Deveau, 1985; Douglas and Peucker, 1973), or the number of coordinates to be considered for conversion to a straight line segment…..(Buttenfield, 1991)*

Some algorithms are extremely simplistic in nature such as the nth point method, while the others are more complex and utilize complex geometric processing. McMaster (1987a) classifies simplification algorithms (Table 3.1) based on the extent of linear search used in the mathematical processing.

Table 3.1 Classification of line simplification algorithms by McMaster (1987a)

[CATEGORY 1]: INDEPENDENT POINT ALGORITHMS
    Do not account for the mathematical relationships with the neighboring coordinate pairs; operate independent of topology.
    Examples: nth point routine
        random-selection of points

[CATEGORY 2]: LOCAL PROCESSING ROUTINES
    Utilize the characteristics of the immediate neighboring points in determining selection/rejection.
    Examples: distance between points
        angular change between points
        Jenks' algorithm

[CATEGORY 3]: CONSTRAINED EXTENDED LOCAL PROCESSING ROUTINES
    Search beyond 'immediate' coordinate neighbors and evaluate sections of the line. Extent of search depends on distance, angular, or number of points criterion.
    Examples: Lang algorithm
        Opheim algorithm
        Johnnsen algorithm
        Deveau algorithm
        Roberge algorithm

[CATEGORY 4]: UNCONSTRAINED EXTENDED LOCAL PROCESSING ROUTINES   Search beyond 'immediate' coordinate neighbors and evaluate sections of the line. Extent of search is constrained by geomorphological complexity of the line, not of algorithmic criterion.

Examples: Reumann-Witkam algorithm
[CATEGORY 5]: GLOBAL ROUTINES Considers the entire line, or specific line segment; in processing. Interactively selects critical points. Examples: Douglas algorithm

All the above-mentioned algorithms except global routines are sequential in nature. While most of the sequential routines operate on the principle of minimizing displacement, a global routine focuses on the careful selection of the critical points, or the salient geometric characteristics of the line (McMaster, 1989). Most of line simplification algorithms require the user to supply a tolerance value, which is used to determine the amount of simplification to be occurred.

Fourier and Wavelet transforms have also been used in line simplification. In these approaches each line is represented by a set of coefficients. The low frequency coefficients characterize the global shapes and have to be kept unchanged while the high frequency coefficients represent the small details and must be cut off (Plazanet, et al., 1996). Filters are used to modify, keep or cut off the coefficients. The coefficients are thresholded, i.e. coefficients whose absolute value is below a selected threshold are set equal to zero. Research has shown some problems in using Fourier and Wavelet transformations. For example with Fourier transformation, some intermediate details show a strong resistance to the smoothing and a drift in the curve location can appear (Plazanet, et al., 1996). Although Wavelet transformation has shown better results compared to Fourier transformation, however, the problems of the location drifts and the shortening of the bends are remained (Plazanet, et al., 1996). Both of these approaches appear not to provide high compression because of the usual randomness in the vector map dataset and simplification is limited to a

certain amount due to the nature of these approaches. Fourier and Wavelet transformations have also been briefly discussed in section 2.3 of chapter 2.

Line simplification approach has some potential advantages compared to transformation techniques, namely the potential possibility of selecting a tolerance value based on the map positional accuracy and keeping the most critical points in their original locations. Douglas-Peucker (1973) line simplification algorithm (sometimes called Douglas algorithm) is one of the most commonly used line simplification algorithms. Douglas-Peucker algorithm is a global routine and recursively processes the entire line in first stage, and then specific subsections of the line. White (1983) calls this algorithm as the most accurate at selecting critical points. McMaster (1986) states that the Douglas-Peucker algorithm is the most cartographically sound routine. Based on a comprehensive analysis of several algorithms, McMaster (1987b) calls this algorithm as one of the most geometrically efficient algorithms in processing strings of x-y coordinate pairs. Additionally, the Douglas-Peucker algorithm's selection of the critical points generates simplifications that mimic those generated by manual generalization, and retains details critical for map-reader recognition (Buttenfield, 1991).

The Douglas-Peucker algorithm (1973) has been selected as line simplification method for this research because of its advantages compared to the other methods. Some modifications will be applied to this algorithm based on map accuracy specifications.

3.3 Douglas-Peucker Line Simplification Algorithm

The Douglas-Peucker algorithm (1973) is one of the most commonly used line simplification algorithms. It is designed to eliminate high frequency details along a line while preserving the overall line shape (Veregin, 2000). It's a global routine and considers the entire line or specific line segments. Initially the endpoints of the line are considered as the start and the end nodes of a straight line, which is called trend line. These two points will remain in the simplified line. Next, the perpendicular distances between each vertex on the line and the trend line is computed and the largest distance is found. If the largest distance is greater than a specified tolerance value, the associated vertex remains in the simplified line and the line is divided to two segments. This process recursively continues until no computed largest distance is greater than the specified tolerance value. At the end, the simplified line contains a subset of vertices constituting the original line. The level of simplification depends on the tolerance value. The result of extreme simplification (with maximum possible tolerance value) is a straight line, which connects the two endpoints of the original line. Figure 3.1 illustrates the Douglas-Peucker algorithm. The highlighted lines illustrate the largest distance greater than tolerance value.

Defining a proper tolerance value is one of the most important issues in using line simplification algorithms including Douglas-Peucker algorithm. Users are usually forced to experiment with different tolerance values to get the desirable results. This is an uncertain process.

Original
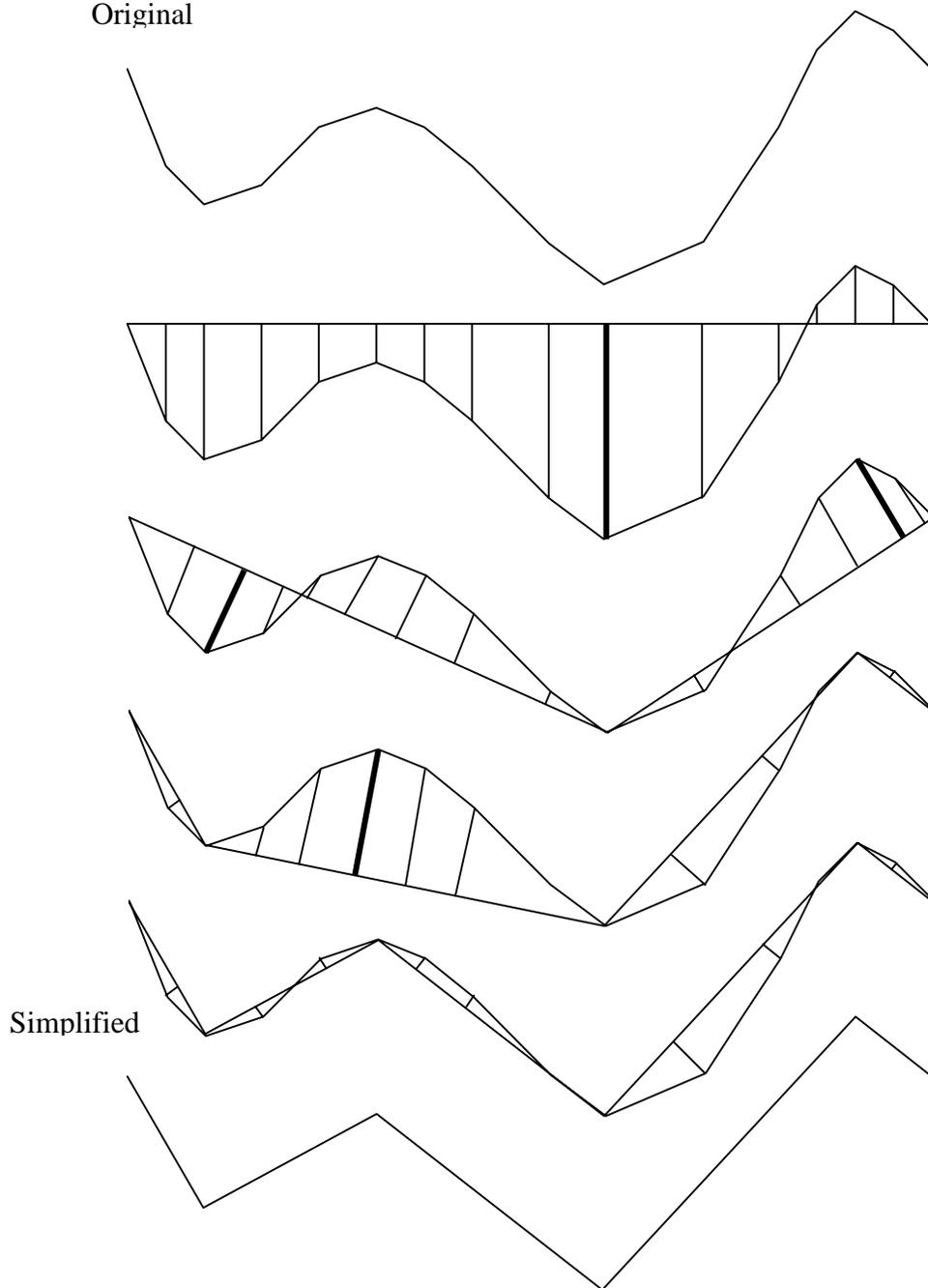
Simplified

Figure 3.1 The Douglas-Peucker Line Simplification Algorithm

3.3.1 Test Data Set

In this research work, Douglas-Peucker algorithm has been implemented using C++ programming language. The input of this program is in ESRI Shape file format and the

output is in binary format. Two linear feature types (roads and rivers) in two different map scales have been selected to be simplified using Douglas-Peucker algorithm. All four data sets are in UTM coordinate system. Table 3.2 shows the characteristics of these four data sets. Datasets 1 and 2 are part of National Atlas of Canada and have been downloaded from "geogratis.cgdi.gc.ca". Datasets 3 and 4 are part of Digital Chart of the World and have been downloaded from "www.GISDataDepot.com".

Table 3.2 Data sets selected to be simplified by Douglas-Peucker algorithm

Data Set #	Feature Type	Location	Map Scale	File Size
1	Road	Canada-Zone12	1:7,500,000	140 KB
2	River	Canada-Zone12	1:7,500,000	467 KB
3	Road	Alberta-Zone 12	1:1,000,000	297 KB
4	River	Alberta-Zone 12	1:1,000,000	1,334KB

3.3.2 Analysis of Simplification Results

All four data sets were simplified using Douglas-Peucker algorithm. The program also calculates the maximum possible tolerance value by which all lines in the map will be simplified to straight lines. Different tolerance values were selected to simplify the maps: 100 meter, 1000 meter, 10,000 meter and maximum possible tolerance value (T_{max}). The compression ratio is calculated using the equation 3.1:

$$(1 - (simplified\_file\_size / original\_file\_size)) * 100 \qquad 3.1$$

For example, if the compression ratio is 75 %, it means that we have subtracted 75 % of the original file. The processing (simplification) time has been calculated in seconds. A

Pentium III computer with 128 MB Ram has been used for these processing. Table 3.3 shows the results of simplification.

Table 3.3 The results of simplification of four map data sets.

Data Set #	Orig.File Size	Maximum possible T	T=100 Comp.Ratio & Time	T=1000 Comp.Ratio & Time	T=10,000 Comp.Ratio & Time	T_{max} Comp.Ratio & Time
1	140 KB	99,490 m	39.29 % 0.38 s	79.29 % 0.17 s	91.43 % 0.14 s	92.14 % 0.14 s
2	467 KB	112,796m	15.42 % 1.19 s	52.68 % 0.55 s	72.16 % 0.39 s	74.09 % 0.35 s
3	297 KB	24,147 m	38.72 % 0.57 s	64.98 % 0.30 s	71.04 % 0.26 s	71.38 % 0.26 s
4	1334KB	34,347 m	22.86 % 2.64 s	61.24 % 1.13 s	66.49 % 0.98 s	66.57 % 0.97 s

As it was expected the compression ratio increases by increasing tolerance value. It means larger tolerance value results in more line simplification. However, the processing time decreases by increasing tolerance value. It means more simplification (larger tolerance value) needs less time. The reason is that by defining larger tolerance value, in each level of simplification, most of the vertices of lines fall into the tolerance boundary and therefore no further simplification is needed. Figure 3.2 illustrates the simplification of a road from data set 1 with two different tolerance values: 1000 m (left) and $T_{max} = 99490$m (right). As it can

be seen from this figure (right), maximum tolerance value results in a straight line connecting the two endpoints of the original line. However, the lower tolerance value results in a line, which is less simplified and more follows the original line.
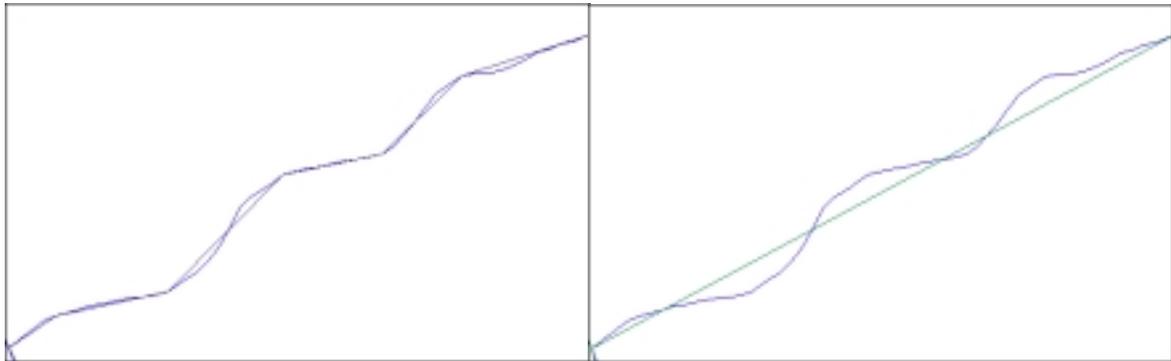
Figure 3.2 Simplification of a road from data set 1 with two different tolerance values: T=1000m (left) and T= 99490m (right).

Now the question is how to select a proper tolerance value for each data set. The compression ratio, as it can be seen from the table 3.3, can be increased to even more than 90% for some data sets, however, it will affect the quality of data regarding positioning accuracy. All simplification algorithms including Douglas-Peucker algorithm induce positional errors in the simplified data, because they produce a discrepancy between the original line and its simplified version. More simplification results in more positional errors. The missing part of the table 3.3 is the amount of positional error induced after simplification. The user defines a tolerance value for simplification but is unaware of the quality of the results. The amount of positional error induced by line simplification algorithms depends on both tolerance value and the complexity of the lines. The geometric complexity of lines within a data set is not the same for all lines. Therefore defining the

same tolerance value for all lines within a data set will result in different amounts of positional errors for different lines. These are very important issues that are missing in most simplification researches.

3.4 Summary

Line simplification is an important function in cartography and GIS. In line simplification source data are transformed to reduce data volume, to merge databases with different scales, or to maintain cartographic quality when scale is reduced. Several line simplification algorithms have been presented by researchers. Most of these algorithms ask user to supply a tolerance value, which is used to determine the amount of simplification to be occurred. Defining a proper tolerance value is one of the most important issues in using line simplification algorithms. All simplification algorithms induce positional errors in data set. The amount of positional error induced by line simplification algorithms depends on both tolerance value and the complexity of the lines. The geometric complexity of lines within a data set is not the same for all lines. Therefore defining the same tolerance value for all lines within a data set will result in different amounts of positional errors for different lines. These very important issues that are missing in much of the line simplification researches will be addressed in the next chapters of this thesis.

Chapter 4

COMPLEXITY MEASURES OF LINEAR FEATURES

4.1 Introduction

Line simplification algorithms selectively eliminate some vertices along a cartographic line to remove unwanted information. Therefore, the simplified line has fewer vertices than the original line. However, this usually results in discrepancy between the geometric position of the original line and the simplified line. This discrepancy is usually referred to as *positional error*.

Most of line simplification algorithms require user to supply a tolerance value, which is used to control the amount of simplification that will take place. Although the user defines a tolerance value, however, the user is unaware of the impact of simplification on positional accuracy of the map. The reason is, mainly, due to the fact that there is no quantitative relation between the tolerance value and the degree of positional errors. It seems logical to assume that the degree of distortion in positional accuracy depends on the geometric complexity of line. The degree of positional error also depends on the simplification algorithm and tolerance value. Each algorithm simplifies the line in a different way, causing different amount of positional errors. Comparing different line simplification

algorithms regarding their impact on positional accuracy of the map is not the goal of this research. Due to its many advantages, the Douglas-Peucker algorithm has been used in this research.

> *Results indicate that the algorithm produces relatively low error levels, especially at high levels of generalization where other methods yield poor results. The algorithm also yields a relatively uniform distribution of error along lines. (Veregin, 2000)*

The amount of positional error highly depends on the geometric complexity of linear features; therefore, investigating the effect of simplification on the geometric complexity of lines is a very important issue. Geometric complexity of linear features changes after simplification and the degree of this change depends on the tolerance value used for simplification. To investigate the effect of line simplification on the geometric characteristics of linear features, the measurement of these characteristics is necessary. The measurement of geometric complexity of linear features and the effect of line simplification on the geometric characteristics of lines is the main focus of this chapter.

4.2 Mathematical Measures of Line Complexity

Several mathematical measures have been proposed by many researchers to evaluate the complexity of linear features, such as mathematical measures proposed by McMaster (1986) and Jasinski (1990). Some of these measures are applied to a single line while the others compare the geometry of a line before and after simplification. There are four main categories of complexity measures, which have been referred by most researchers working on complexity measures. These categories of complexity measures are *length*, *density*,

angularity and *curvilinearity* measures (McMaster, 1986). These categories are discussed in the following subsections.

4.2.1 Length Measures

The length of a line is probably the simplest geometric attribute of it. This attribute cannot solely represent the complexity of a line. However, it can be used to investigate the effect of the simplification algorithm on the line. A line becomes shorter, as it undergoes the simplification because it loses its sinuosity (curviness). McMaster (1986) introduced the "percentage change in line length" as a length measure. This measure has been defined as the length of the simplified line divided by the length of the original line in form of a percentage (McMaster, 1986):

$$\sum_{i=1}^{m-1} sls_{(i)} / \sum_{j=1}^{n-1} slo_{(j)} *100 \qquad\qquad 4.1$$

Where *m* is the number of coordinate pairs on the simplified line, *n* is the number of coordinate pairs on the original line, *sls* is the individual segment lengths on the simplified line and *slo* is the individual segment lengths on the original line. More simplification decreases the value of this ratio.

Jasinski (1990) presented another length measure, "error variance", which is an average perpendicular displacement of every point in a line to the anchor line. In fact error variance measures the deviation of a line from its straight-line approximation, which is called anchor line. Buttenfield (1984) defined the anchor line as a straight segment linking the first and the last point of the coordinate string. Error variance is expected to increase as the line is

more simplified. This measure is based on a mean value. Mean value of a distribution, in general, does not provide the dispersion of values in the whole data set. Therefore, Jasinski (1990) introduced another length measure, called the "coefficient of variation of error variance". Coefficient of variation calculates the relative variability of a distribution as a ratio of the standard deviation to the mean value.

4.2.2 Density Measures

Density of a line represents the frequency of details in the line. Number of points of a line, as a measure of density, solely cannot represent the complexity of the line. However, it can be used to represent the effect of simplification on the line. All simplification algorithms, principally, reduce the number of points along a line. Hence, density of a line is expected to decrease as the line is more simplified.

Jasinski (1990) introduced a density measure called "average segment length" which is the average length of all segments between the points of the line. Although this measure is based on the length, however, it represents the density of a line. The value of this measure is expected to increase as a line is more simplified because fewer points are retained and segments automatically get longer. Jasinski (1990) also introduced another density measure called "coefficient of variation of average segment length" because the average segment length does not give much information whether most of the segments are similar or whether the line consists of some very short segments and some very long ones.

McMaster (1986) introduced three density measures to compare the density of a line before and after simplification. These measures are:

- Ratio of the change in number of coordinates,

- Difference in average number of coordinates per inch, and

- Ratio of the change in the standard deviation of the number of coordinates per inch.

The "ratio of the change in number of coordinates" has been defined as number of points of the simplified line divided by the number of points of the original line in form of a percentage (McMaster, 1986):

$$m/n*100 \qquad\qquad\qquad 4.2$$

Where m is the number of points of the simplified line and n is the number of points of the original line. More simplification decreases the value of this ratio.

4.2.3 Angularity Measures

Angularity of a line is one of its primary geometric characteristics. Angularity measures evaluate specifically the individual angular changes along a line. Jasinski (1990) defined two angularity measures called "average angularity", which ranges from 0 (straight lint) to 1, and "coefficient of variation of average angularity". The latter is expected to decrease as the line is more simplified.

McMaster (1986) defined nine measures for angularity of a line to compare its angularity before and after simplification. The "percentage change in angularity" is one of these measures, which was expressed as the sum of the angles between consecutive vectors on the simplified line divided by this sum on the original line:

$$ABS \sum_{i=1}^{m-1} angs_{(i)} \, / \, ABS \sum_{j=1}^{n-1} ango_{(j)} *100 \qquad\qquad 4.3$$

Where *m* is the number of coordinate pairs on the simplified line, *n* is the number of coordinate pairs on the original line, *angs* is the angle of change between two consecutive vectors on the simplified line and *ango* is the angle of change between two consecutive vectors on the original line.

Plazanet et. al. (1996) measure the geometric characteristics of individual bends of a line segment. Then, the authors classify line segments based on the mean (or median) value, variance, minimum and maximum values of each of these measurements. The authors define bend as a fraction of curve between two consecutive inflection points. Inflection points divide the line to curvilinear segments, which are the portions of a line in which all angles are in the same direction, either positive or negative.

4.2.4 Curvilinearity Measures

Curvilinearity of a line is defined by the number of inflection points in the line. Curvilinearity is the measure of direction of angular changes while angularity is the measure of magnitude of angular changes. Jasinski (1990) defined the ratio of total curvilinearity to the number of all turns as the "curvilinearity ratio". McMaster (1986) defined four curvilinearity measures. The "ratio of the change in the number of curvilinear segments" is one of these measures, which was expressed as the number of curvilinear segments of the simplified line divided by the number of curvilinear segments of the original line:

$$a/b*100 \hspace{6cm} 4.4$$

Where *a* is the number of curvilinear segments on the simplified line and *b* is the number of curvilinear segments on the original line.

4.3 Discussion of Data and Measurements

A set of four lines was selected for complexity measures. Each of these lines is a part of a river in the United States. This data has a scale of 1:3,000,000 and has come with ESRI ArcView 3.1 software. For simplicity, the rivers have been named river1, river2, river3 and river4, which are parts of the rivers "Platte", "Arkansas", "Brazos" and "Red River of the North" respectively. In order for the set of lines to be representative of a wide range of shapes, it was decided to select the rivers with different irregularities. River1, which is less irregular, has 459.46 km length and includes 19 points. The length of river2 is 714.99 km and includes 40 points. River3 is 962.17 km and has 87 points. River4 which is 443.72 km, has almost the same length as river1, but much more irregularities, and includes 137 points. Figure 4.1 illustrates the selected rivers.
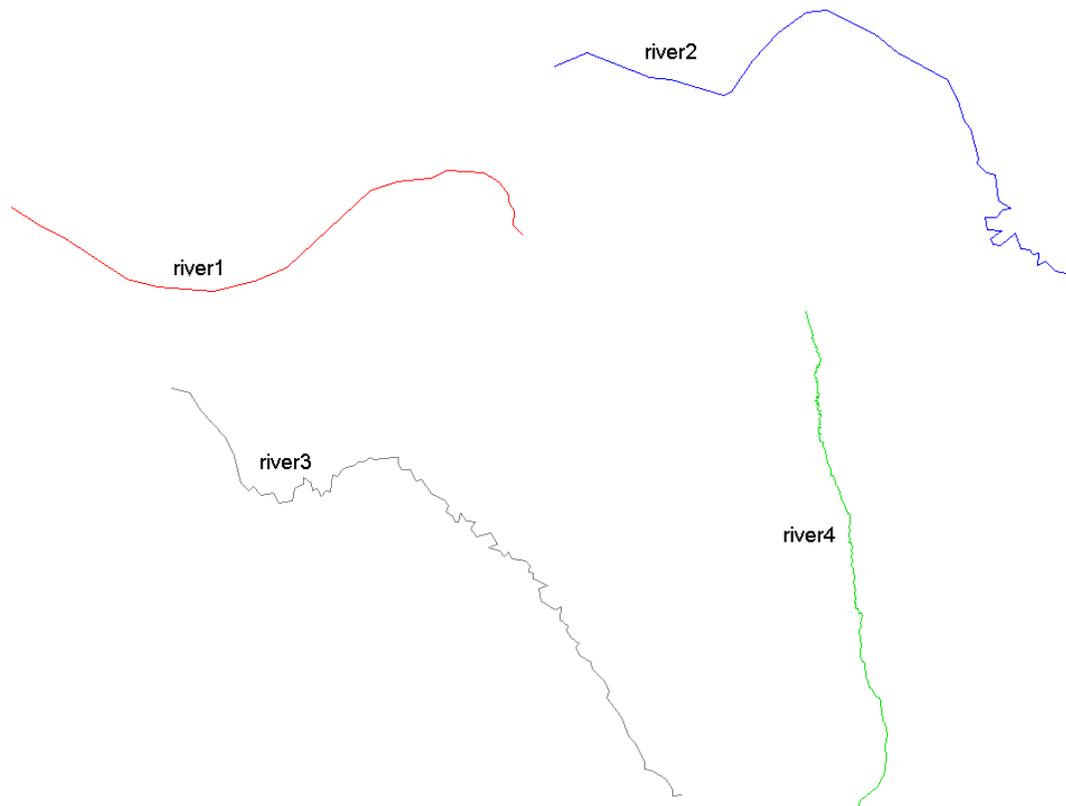
Figure 4.1 Data set used for complexity measures

The geometric characteristics of each of these four lines were measured before and after simplification with different tolerance values. Douglas-Peucker algorithm was used for simplification of the lines. Each aspect of line complexity - namely: length, density, angularity and curvilinearity - is measured. McMaster (1986) measures (equations 4.1, 4.2, 4.3 and 4.4) were used to measure and compare the geometric characteristics of each line before and after simplification with different tolerance values. The results have been plotted as graphs with the values of measures against the progression of the tolerance value of simplification.

The first measure applied to the data set was the "percentage change in line length" which is the length of the simplified line divided by the length of the original line in form of a percentage (equation 4.1). The "percentage change in line length" is a length measure. Figure 4.2 illustrates the results of this measure.
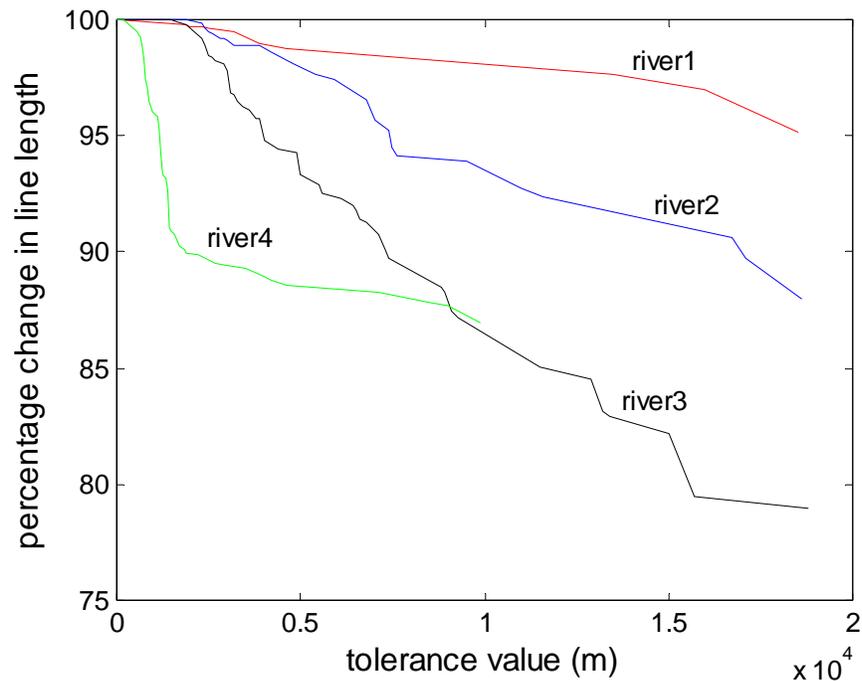
Figure 4.2 Percentage change in line length due to the simplification

As expected, the percentage change in line length is decreasing as the lines are more simplified. The rate of this decreasing depends on the complexity of the line. In the case of river4, the percentage change in line length changes faster in lower tolerance values with respect to the higher values, because its points are close to its anchor line and are removed with low tolerance values. However in the case of river1, which is not very simplified with low tolerance values, the percentage change in line length changes very slowly in lower

tolerance values. Figure 4.2 shows how simplification effects differently on the length of lines with different geometric characteristics.

Figure 4.3 illustrates the results of the second measure (equation 4.2), "ratio of the change in number of coordinates", which is number of points of the simplified line divided by the number of points of the original line in form of a percentage. The "ratio of the change in number of coordinates" is a density measure.
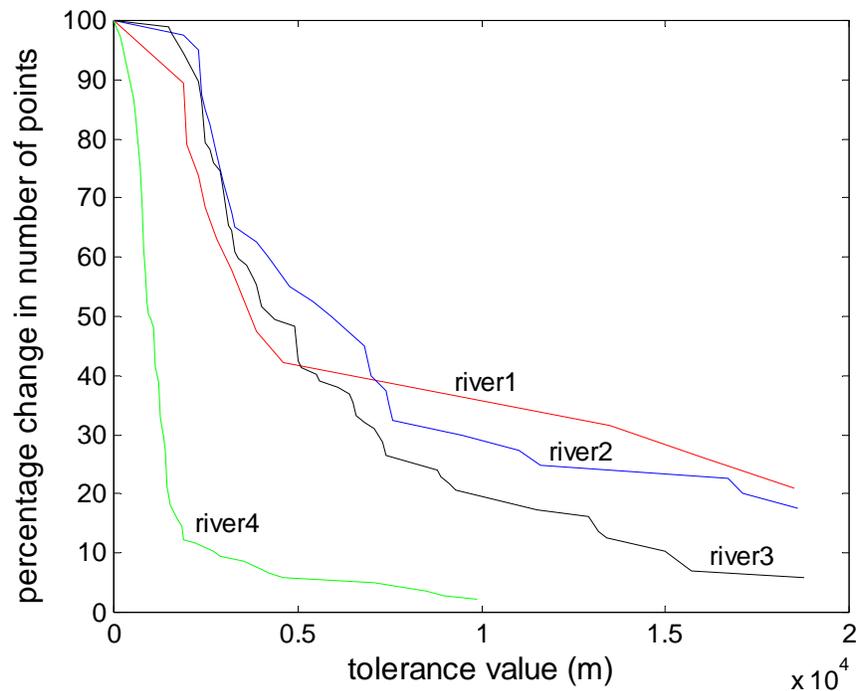
Figure 4.3 Percentage change in number of points due to the simplification

As expected, the percentage change in number of points is decreasing with more simplification. This decreasing is faster in the case of river4, because despite of its high irregularity, its points are close to its anchor line. Therefore even with small tolerance

value, many points are removed by simplification. Figure 4.3 shows how simplification effects differently on the density of lines with different shapes. For example if we use $0.2x10^4$ m as tolerance value for simplification, river4 will lose more than %90 of its points, while the other three lines will lose less than %40 of their density.

The third measure applied to the data set was the "percentage change in angularity" which is the sum of the angles between consecutive vectors on the simplified line divided by this sum on the original line in form of a percentage (equation 4.3). The "percentage change in angularity" is an angularity measure. Figure 4.4 illustrates the results of this measure.
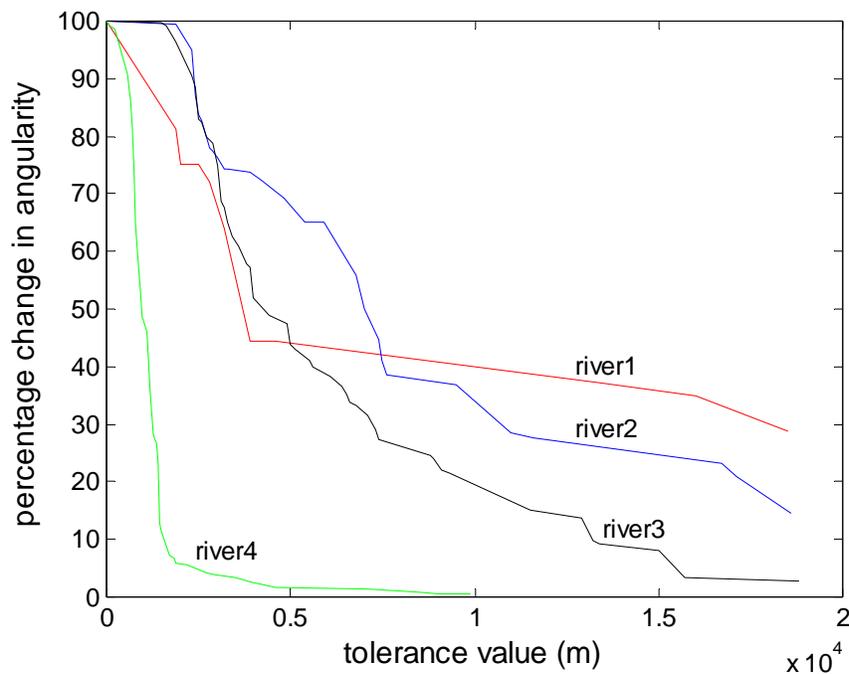

Figure 4.4 Percentage change in angularity due to the simplification

As expected, the percentage change in angularity is decreasing as the lines are more simplified. This decreasing is faster in the case of river4, because this line is mostly

simplified in low tolerance values. Therefore it loses its angularity much faster than the other three lines. Figure 4.4 shows how simplification effects differently on the angularity of lines with different geometric characteristics. For example, if we use $1x10^4$ m for simplification, river4 will lose %100 of its angularity (i.e. it will become a straight line), while the other three lines will keep more than about %20 of their angularity.

Figure 4.5 illustrates the results of the last measure, "ratio of the change in the number of curvilinear segments", which is the number of curvilinear segments of the simplified line divided by the number of curvilinear segments of the original line (equation 4.4). The "ratio of the change in the number of curvilinear segments" is a curvilinearity measure.
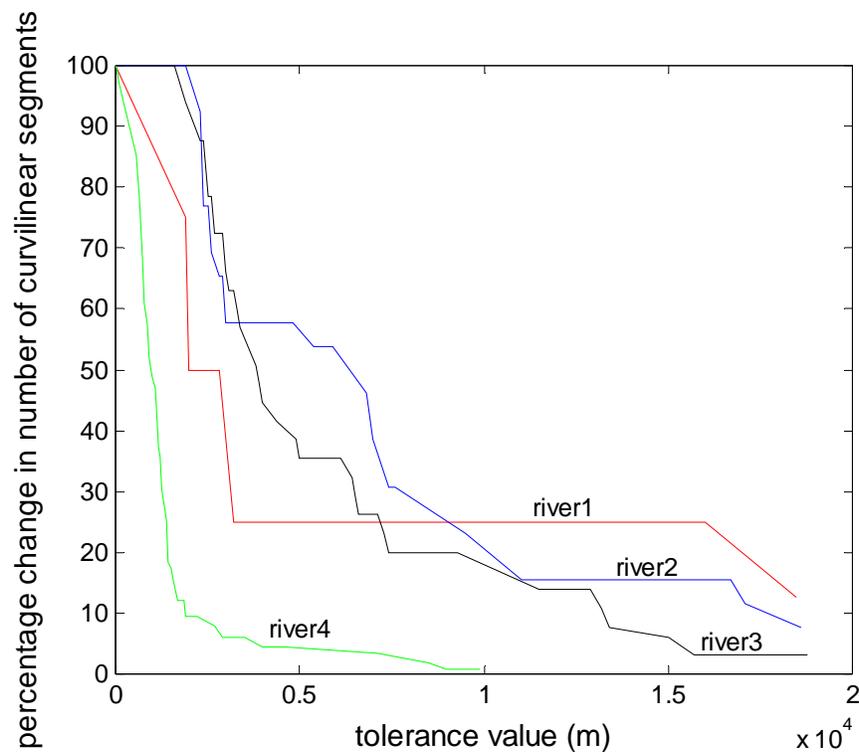

Figure 4.5 Percentage change in curvilinearity due to the simplification

As expected, the percentage change in curvilinearity is decreasing with more simplification. Figure 4.5 shows how simplification effects differently on the curvilinearity of lines with different shapes. For example, using $0.2x10^4$ m as tolerance value, river4 will keep only about %10 of its curvilinearity, while river2 and river3 will keep %100 of their curvilinearity and river1 will keep %50 of its curvilinearity. This is because of different geometric characteristics of these linear features. The horizontal straight lines in figure 4.5 show that the line is more simplified, however, the number of curvilinear segments doesn't change.

All four complexity measures, (length, density, angularity and curvilinearity measures) showed that line simplification effects differently on the complexity of lines with different shapes. In the previous figures, it is clear that the percentage change in the geometric characteristics of the lines is different when we use a single tolerance value to simplify lines with different shapes. This arises the question of whether it is correct to use a single tolerance value to simplify a data set, which includes lines with completely different geometric complexity.

4.4 Summary

All line simplification algorithms create a discrepancy between the geometric position of the original line and the simplified line. This discrepancy is usually referred to as positional error. The amount of positional error highly depends on the geometric complexity of lines; therefore, investigating the effect of simplification on the geometric complexity of lines is a very important issue. This chapter focused on the measurement of geometric complexity of

lines and the effect of line simplification on the geometric characteristics of the lines. The results showed that the geometric characteristics of a line changes due to the simplification. The amount of this change depends on the shape of the line and the tolerance value specified by the user. All results clearly indicate that line simplification effects differently on the complexity of lines with different shapes. Therefore using a single tolerance value for lines with different complexities in a data set is not practical.

Chapter 5

A NEW APPROACH TO MINIMIZE THE POSITIONAL ERRORS

5.1 Introduction

Line simplification algorithms introduce positional errors in the data set. The amount of positional errors depends on the simplification algorithm and the tolerance value used for simplification. There is a relation between the tolerance value and the amount of positional error. Larger tolerance value causes more simplification; therefore, more discrepancy is introduced between the simplified line and the original line and hence more positional error is introduced. The test results of chapter four clearly showed that line simplification algorithm have different effects on the geometric properties of lines with different shapes. Therefore, the geometric characteristic of a line is an important factor in relation between tolerance value and positional error induced by simplification.

In most line simplification algorithms including Douglas_Peucker, users are asked to supply a tolerance value to determine the amount of required simplification. Usually a proper tolerance value is achieved by trial and error. User enters different tolerance values, inspects the visual quality of the results of simplification and selects the most appropriate one. However, the problem is that when a user provides a tolerance value, all lines in the

data set are simplified with the same tolerance value. As the geometric complexity of lines in the data set are not the same, different amounts of positional errors are introduced for different lines. In fact the user doesn't know the amount of error introduced for each line. Still, there is no way for users to identify a tolerance value by which maximum simplification is achieved while attaining a specific level of positional accuracy for all lines in the data set.

> *What is missing is a way for users to select a bandwidth value that maximizes the number of vertices eliminated subject to the constraint that the set of simplified lines maintains a certain level of positional accuracy. Such a goal can only be achieved by trial and error. A better implementation would allow users to specify the acceptable maximum level of error and harness the power of the system to determine the bandwidth required to achieve this goal. (Veregin, 2000)*

What is usually important for user, is to simplify the data set while maintaining a specific level of quality, not the tolerance value itself. The questions are therefore, how to specify a tolerance value for each line based on the user specified level of accuracy and how to apply line simplification using a specified tolerance value for each line rather than using the same tolerance value for the all lines in the data set. The objective is to apply the largest tolerance value for each line that achieves the target level of positional accuracy.

This chapter focuses on positional errors introduced by line simplification algorithms and presents a new approach to determine the tolerance value for each line based on the user specified level of positional accuracy. It will be shown that the new approach minimizes the positional errors introduced by line simplification while maintaining the positional accuracy at the user specified level.

5.2 Positional Errors Introduced by Line Simplification

Line simplification algorithms introduce a discrepancy between the geometric location of the original line and the simplified line. This effects the quality of data. Users can visually compare the quality of the simplified data and the original data. However, this is not enough concerning the data quality standards. The discrepancy needs to be quantified with some measures to ensure that a specific positional accuracy is maintained after simplification. Several measures of positional error have been introduced by researches. Veregin (2000), has defined some of these measures as follows:

- *Distortion polygons*: Measures based on the polygons formed between the original and simplified line (McMaster 1987a, b, Buttenfield 1991).

- *Displacement vectors*: Measures based on perpendicular lines drawn between the simplified and original lines (McMaster 1987a, b, Jenks 1989).

- *Critical distance*: Measures based on the proportion of vertices on the original line that are more than a specified distance threshold away from the simplified line (Little 1989).

McMaster (1986) defined three groups of measures to evaluate positional differences between the simplified line and the original line. He referred to them as displacement or comparative measures. These three measures are "vector displacement", "polygonal displacement" and "perimeter displacement" measures. He also introduced five measures for vector displacement. All these measures are based on perpendicular lines drawn from each eliminated vertex of the original line to the simplified line. He referred to these lines

as vectors between the two lines. Figure 5.1 illustrates a line and its simplified version and the difference vectors between the two lines.
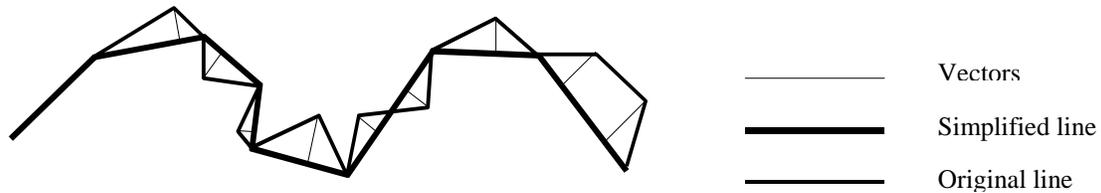
Figure 5.1 Difference vectors between original line and simplified line

One of the vector displacement measures introduced by McMaster (1986) has been used in this research to measure the discrepancy between the original line and its simplified version. This measure is called "total length of vector differences per inch of line" and is calculated as the sum of the length of all vectors between the two lines divided by the length of the original line (McMaster, 1986). This measure is expected to increase with more line simplification. This makes sense because more simplification removes more vertices from the original line and causes more discrepancy between the original line and its simplified version. Therefore the sum of the length of all vectors increases and hence the value of vector displacement increases.

5.2.1 Test Results and Evaluation

In order to investigate the effect of line simplification on the positional accuracy of the lines, vector displacement has been measured after simplification of the four lines discussed in section 4.3 (Shahriari and Tao, 2002). Each line has been simplified with several tolerance values and the vector displacement has been measured each time. Figure 5.2 illustrates the results of these measurements. It can be seen in the graph that the amount of

vector displacement for each line is increasing with the increase of the tolerance value. In the other words, as it was expected, the positional errors increase with more simplification.
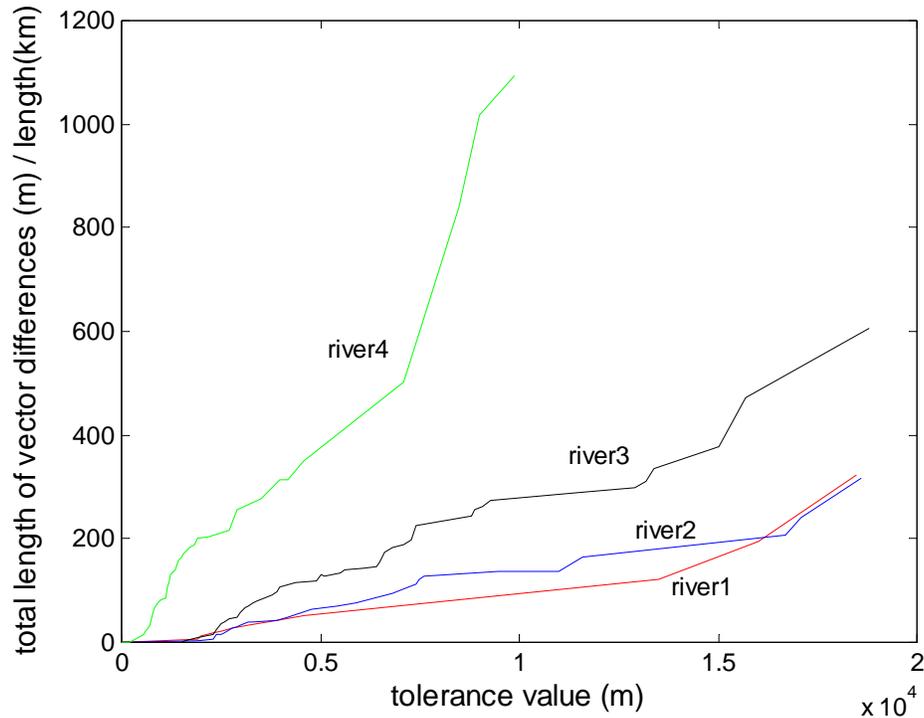
Figure 5.2 Vector displacements due to the simplification (Shahriari and Tao, 2002)

Figure 5.2 shows that the amount of vector displacement is different for each line when using one tolerance value for simplification of all four lines. In most parts of the graph for each specific tolerance value, there are four different values for vector displacement. For example, if specified tolerance is 7000 meters, the vector displacements will be 53.05, 101.25, 181.25 and 350.59 meters per kilometer of line length for river1, river2, river3 and river4 respectively. The reason for obtaining different results is the difference in the geometric complexity of lines, which was discussed in Chapter 4. If specified tolerance value is small, for example 2000 meters, no significant positional errors will be introduced

in river1, river2 and river3. However, in the case of river4, considerable amount of positional error would be introduced. This is due to the significant difference in its geometric complexity compared to the other three lines. This is very clear now, that a specific level of positional accuracy cannot be guaranteed when using the same tolerance value to simplify all lines in a data set. A new approach will be introduced in the next section, by which each line is simplified using a specific tolerance value.

5.3 A New Approach to Maintain a Specific Level of Positional Accuracy

Line simplification, as discussed above, can have significant effects on the data quality; however, the user doesn't have any control on these effects. User supplies a tolerance value to simplify all lines in the data set, but is unable to specify a target level of positional accuracy to be maintained after simplification. It was shown in the previous section that using one tolerance value for simplification of all lines in a data set introduces different amounts of positional errors for different lines. Therefore, in order to keep a certain level of positional accuracy for all lines in a data set after simplification, different tolerance values should be used for different lines. The ideal case is that the user specifies a target level of positional accuracy or maximum acceptable positional error and then the tolerance value for each line is estimated based on user specified level of accuracy. Now the questions are how to estimate a tolerance value for each line based on the user specified level of accuracy and how to simplify each line using its estimated tolerance value. In other words, we seek a way to calculate maximum tolerance value for each line by which the user specified positional accuracy is maintained and then simplify each line with this estimated tolerance value. Shahriari and Tao (2002) present a new way to solve this problem. In this approach, the user supplies a certain level of positional accuracy (either vector displacement or

polygonal displacement or perimeter displacement or any other measure of positional accuracy depending on the user application) instead of supplying a tolerance value. Then the proper tolerance value is automatically specified for each line in the data set. The proper tolerance value for each line is a value by which maximum simplification is provided while maintaining the user specified positional accuracy. In order to obtain this value for each line, the data set has to be pre-analyzed. In this pre-analysis phase, positional displacement measure is applied to all lines in the data set using a set of tolerance values, which starts from zero and ends by the maximum possible tolerance value for each line. Maximum possible tolerance value for each line is the value by which the line is simplified to a straight line connecting two end nodes of the line. The set of tolerance values and the corresponding positional displacements along with the feature identifier (a unique number to identify each line in a map) of all lines in the map are, then, recorded in a single text file. After this pre-analysis phase, simplification is done using the text file created in the pre-analysis phase. Users enter a value as maximum valid positional displacement, then, the proper tolerance value for each line is found from the text file. The proper tolerance value for each line is the maximum tolerance value that has an associated positional displacement less than the user specified level of positional displacement. As each line is simplified using its adapted tolerance value, this approach can be called "line simplification using adaptive tolerance values". In this research vector displacement, which is the total length of vector displacements divided by the length of the original line, has been used as positional displacement measure. An example of this process is graphically shown in figure 5.3. In this example, the user has specified 150m per km as the maximum vector displacement and the values of 14438.86 m, 11271.90 m, 6436.25 m and 1398.41 m have been calculated as

tolerance values for river1, river2, river3 and river4 respectively. Therefore, the four rivers will be simplified using four different tolerance values while the maximum vector displacement will remain under 150 m per km for all four lines.
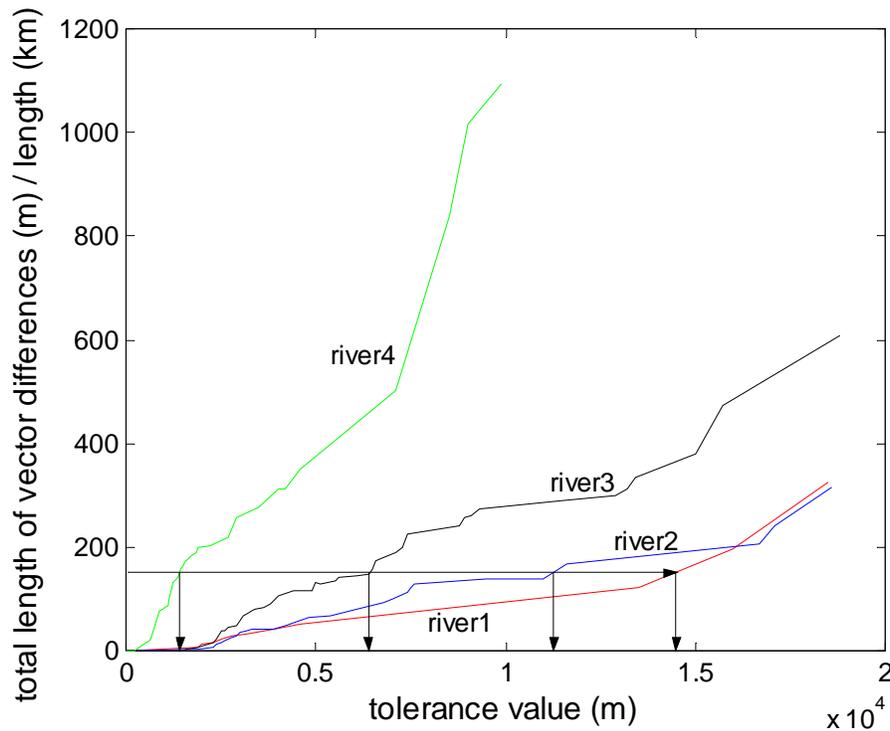
Figure 5.3 Adaptive tolerance values (Shahriari and Tao, 2002)

The result of simplification on river4 using the user specified tolerance value is shown in the left side of figure 5.4 and the result of simplification using the adaptive tolerance value is shown in the right side of figure 5.4. The figures show only a part of this river. As it was expected the vector displacement is smaller when using the adapted tolerance value.
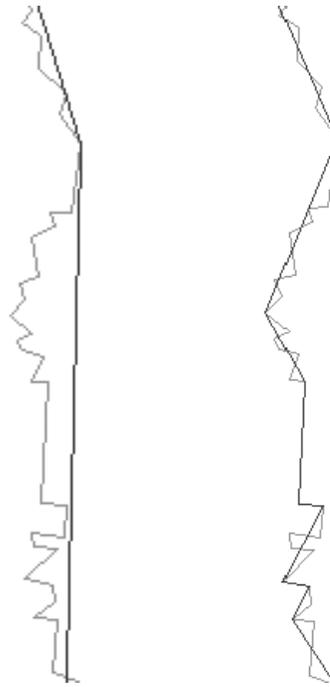
Figure 5.4 Simplification with non-adaptive tolerance value (left) and with adaptive tolerance value (right)

5.3.1 Test Results and Evaluation

This approach has also been tested on a larger data set. C++ programming language has been used to program both pre-analysis phase and simplification phase. The first test data set is data set #1 (refer to Table 3.2), which is a shape file containing Canada rivers located in UTM zone 12. The size of this shape file is 467 KB and includes 1353 lines. The first step, as mentioned before, is the pre-analysis phase in which the vector displacements are calculated for each line using different tolerance values. In order to specify a set of tolerance values for each line, first maximum possible tolerance value (T_{max}) for that line (by which the line will be simplified to a straight line connecting the start point and end point of the line) is calculated. Then a set of tolerance values is selected between zero and T_{max}. Experience on different data sets showed that most of the lines get very close to their

maximum simplification by tolerance value equal to $0.5*T_{max}$. Therefore, almost all of the tolerance values are selected between zero and $0.5*T_{max}$ by dividing this range to 60 equal intervals. The values of $0.75*T_{max}$ and T_{max} are also selected. After assigning tolerance values for each line, vector displacement of the line is calculated for each tolerance value in the set of assigned tolerance values for that line. It should be noted that the lines of the map are not actually simplified in this pre-analysis phase. However, the values of vector displacements are calculated based on the question "what would be the value of vector displacement if the line was simplified with a specific tolerance value?". All the assigned tolerance values and corresponding vector displacements along with the feature identifiers are automatically recorded in a text file. Therefore the result of the pre-analysis phase for a map is a single text file. It also should be noted that the program doesn't let any repeated pair of tolerance value and associated vector displacement to be recorded in this text file. In other words, if the calculated vector displacement for a specific tolerance value is equal to the vector displacement of the previous tolerance value, it will not be recorded as a new pair in the text file. Table 5.1 shows the structure of this text file.

Table 5.1 The file structure of pre-analysis result

Structure		Example	
ID #		1	
Part #		1	
0	0	0	0
T_1	VD_1	255.101	20.7483
T_2	VD_2	510.203	41.4608
T_3	VD_3	765.304	58.494
…..		…..	
…..		…..	
T_{max}	VD_{max}	1530.61	111.425
-2		-2	
Part #		2	
0	0	0	0
T_1	VD_1	259.443	17.7157
T_2	VD_2	518.887	62.4915
T_3	VD_3	1037.77	91.8747
…..		…..	
…..		…..	
T_{max}	VD_{max}	2594.43	304.112
-2		-2	
-1		-1	
ID #		2	
……		……	

As it can be seen from table 5.1, the text file starts with feature identifier number for the first feature and then continues with part number (lines are divided into parts at nodes in ESRI shape file format). Then it is followed by the list of pairs of tolerance value and associated vector displacement value for that part. Tolerance value starts from zero and ends to maximum possible tolerance value for that part. Number -2 indicates the end of the list for a part and number –1 indicates the end of a list for a line.

The next step after pre-analysis is the actual simplification using adaptive tolerance values. For this step, the value of 100 m per km was specified as the maximum valid vector

displacement for all the lines of the map (as user specified level of positional accuracy). Then the proper tolerance value for each line is automatically selected using the text file created in the pre-analysis phase. The proper tolerance value for each line, in this test, is the maximum tolerance value that has an associated vector displacement less than 100 m per km. Next, each line of the map was simplified using the specified tolerance value which is referred to as adaptive tolerance value. To compare the result of the adaptive algorithm with those simplification with a constant tolerance value for all lines of the map, the original data set (data set #1) was simplified using the tolerance value of 1500m. Figure 5.5 shows a section of this data set (top), together with the results of using constant tolerance value (1500 m) and the result of using adaptive tolerance value (100 m/km). As expected, the amount of vector displacement induced by line simplification is completely different for different lines in the case of using constant tolerance value (Figure 5.5, left). However, the vector displacement has been kept under 100 m per km for all lines in the case of simplification using adaptive tolerance values (Figure 5.5, right). This means that user has control on the maximum vector displacement induced by line simplification. The processing time for the case of constant tolerance value was 0.601 s while for the adaptive case was 0.911 s plus 13.309 s for the pre-analysis phase. The reason that more processing time is needed for adaptive approach is partly because of searching the text file to find the proper tolerance value for a line. As the order of the lines in the text file is the same as their order in the map file, the program does not have to search whole text file to find the proper tolerance value for one line but only the section of the file that starts with the identifier of that line.
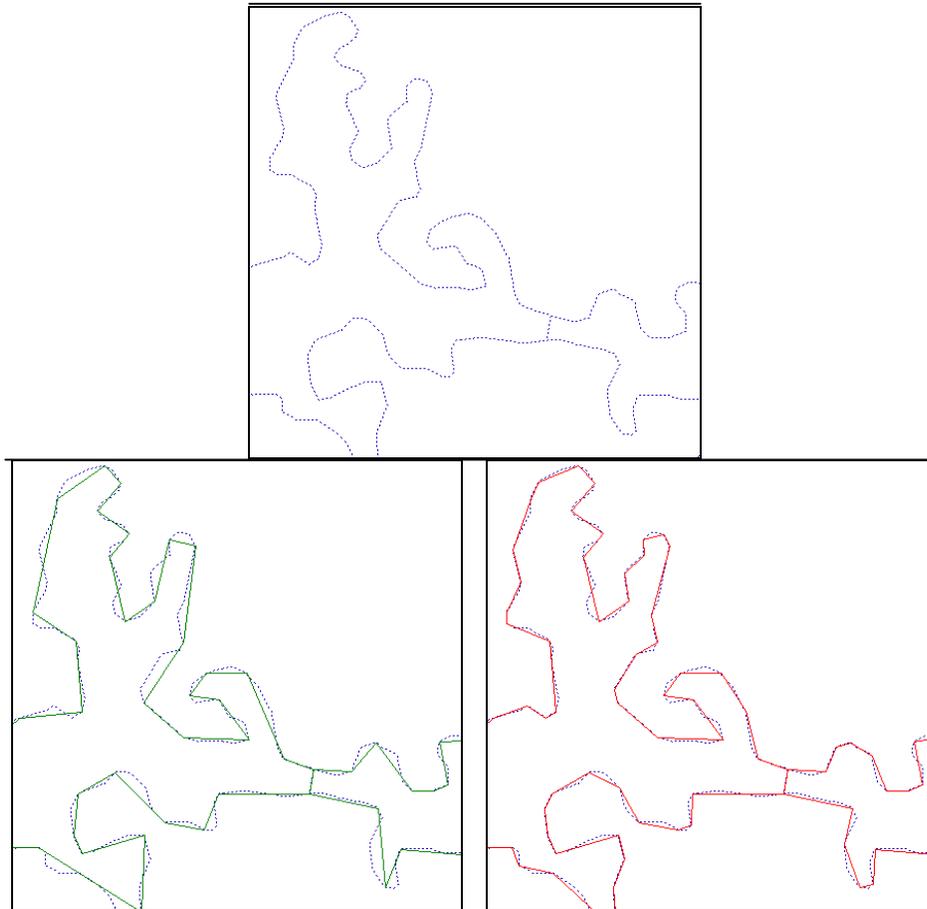
Figure 5.5 A section of the test data set (up), the results of using constant tolerance value (left) and the result of using adaptive tolerance value (right)

(Shahriari and Tao, 2002).

This new approach has also been tested on the other test data sets including roads (man-made features) and rivers (natural features) in different scales. These data sets are listed in Table 3.2 and Table 5.2. A Pentium III computer with 128 MB Ram and 550 MHz CPU speed has been used for all the processing. Datasets 5 and 6 are part of NTDB of Canada (082F04, Edition 4, 1999/01/05).

Table 5.2 Additional data sets used to test the new algorithm

Data Set #	Feature Type	Location	Map Scale	File Size
5	Road	Vancouver-Zone11	1:50,000	3,627KB
6	River	Vancouver-Zone11	1:50,000	2,884KB

Tables 5.3, 5.4, 5.5, 5.6, 5.7 and 5.8 list the result of these tests. Each table corresponds to one of the data sets listed in Table 3.2 and Table 5.2. In the case of constant tolerance value, the user specifies a tolerance value to simplify all lines in the data set with the same value. In the case of adaptive approach, however, the user enters a maximum valid vector displacement as a positional accuracy criterion for simplification. Maximum vector displacement after simplification has been mentioned for each case in the tables. The time needed for simplification and achieved compression ratio for each case have also been mentioned in the tables. The time needed for pre-analysis phase for adaptive approach has been mentioned in each table's caption. In adaptive approach, pre-analysis phase is done only once for each data set; however, user can simplify the data set using different values of vector displacement as accuracy level.

Table 5.3 Test results on data set#1 (Pre-analysis time for adaptive approach= 5.6 s)

Tolerance Value	User Input	Max. Vec. Disp.	Time	Comp. Ratio
Constant	T= 100m	36.0 m/km	0.38 s	39.3 %
Adaptive	VD= 18 m/km	18.0 m/km	0.50 s	39.8 %
Constant	T= 1000m	398.4 m/km	0.17 s	79.3 %
Adaptive	VD= 130 m/km	129.9 m/km	0.30 s	79.6 %
Constant	T= 10000m	3315.2 m/km	0.14 s	91.4 %
Adaptive	VD=1650m/km	1642.4 m/km	0.26 s	91.4 %
Constant	T_{max}= 99490 m	16902.0 m/km	0.14 s	92.1 %
Adaptive	VD=4500m/km	4493.0 m/km	0.26 s	91.8 %

Table 5.4 Test results on data set#2 (Pre-analysis time for adaptive approach =16.7 s)

Tolerance Value	User Input	Max. Vec. Disp.	Time	Comp. Ratio
Constant	T= 100m	40.7 m/km	1.19 s	15.4 %
Adaptive	VD= 4 m/km	4.0 m/km	1.53 s	15.5 %
Constant	T= 1000m	425.9 m/km	0.55 s	52.7 %
Adaptive	VD= 113 m/km	112.9 m/km	1.03 s	52.8 %
Constant	T= 10000m	2750.2 m/km	0.39 s	72.2 %
Adaptive	VD=1400m/km	1399.3 m/km	0.82 s	72.3 %
Constant	T_{max}=112796 m	20106.6 m/km	0.35 s	74.1 %
Adaptive	VD=6500m/km	6435.9 m/km	0.79 s	73.8 %

Table 5.5 Test results on data set#3 (Pre-analysis time for adaptive approach=10.2s)

Tolerance Value	User Input	Max. Vec. Disp.	Time	Comp. Ratio
Constant	T= 100m	62.8 m/km	0.57 s	38.7 %
Adaptive	VD= 4m/km	4 m/km	0.74 s	38.7 %
Constant	T= 1000m	626.8 m/km	0.30 s	65.0 %
Adaptive	VD= 190 m/km	189.5 m/km	0.63 s	65.0 %
Constant	T= 10000m	3047.4 m/km	0.26 s	71.0 %
Adaptive	VD=1300m/km	1264.2 m/km	0.59 s	71.0 %
Constant	T_{max}= 24147 m	6517.0 m/km	0.26 s	71.4 %
Adaptive	VD=4300m/km	4229.2 m/km	0.58 s	71.3 %

Table 5.6 Test results on data set#4 (Pre-analysis time for adaptive approach =42.3 s)

Tolerance Value	User Input	Max. Vec. Disp.	Time	Comp. Ratio
Constant	T= 100m	228.0 m/km	2.64 s	22.9 %
Adaptive	VD= 8 m/km	7.3 m/km	3.56 s	22.9 %
Constant	T= 1000m	1359.3 m/km	1.13 s	61.2 %
Adaptive	VD= 445 m/km	444.7 m/km	2.35 s	61.2 %
Constant	T= 10000m	7770.5 m/km	0.98 s	66.5 %
Adaptive	VD=5300 m/km	5299.7 m/km	2.19 s	66.3 %
Constant	T_{max}=34347 m	19883.3 m/km	0.97 s	66.6 %
Adaptive	VD=16000m/km	15903.1m/km	2.18 s	66.5 %

Table 5.7 Test results on data set#5 (Pre-analysis time for adaptive approach=126.2 s)

Tolerance Value	User Input	Max. Vec. Disp.	Time	Comp. Ratio
Constant	T= 10 m	908.4 m/km	4.66 s	76.8 %
Adaptive	VD= 200m/km	200.0 m/km	7.44 s	76.6 %
Constant	T= 100 m	4056.5 m/km	4.26 s	82.8 %
Adaptive	VD= 2600m/km	2598.3 m/km	6.89 s	82.8 %
Constant	T= 500 m	16217.5 m/km	4.05 s	83.3 %
Adaptive	VD= 6000m/km	5997.0 m/km	6.84 s	83.0 %
Constant	T_{max}= 943 m	30244.3 m/km	3.92 s	83.4 %
Adaptive	VD= 7000m/km	6993.6 m/km	6.80 s	83.2 %

Table 5.8 Test results on data set#6 (Pre-analysis time for adaptive approach =105.2 s)

Tolerance Value	User Input	Max. Vec. Disp.	Time	Comp. Ratio
Constant	T= 10 m	583.5 m/km	3.85 s	81.0 %
Adaptive	VD= 210 m/km	210.0 m/km	5.86 s	80.9 %
Constant	T= 100 m	6152.6 m/km	3.34 s	87.9 %
Adaptive	VD=3000 m/km	2994.5 m/km	5.76 s	87.9 %
Constant	T= 500 m	19021.8 m/km	3.04 s	88.2 %
Adaptive	VD=5000 m/km	4994.7 m/km	5.20 s	88.0 %
Constant	T_{max}= 908 m	31026.3 m/km	3.03 s	88.3 %
Adaptive	VD=7000m/km	6981.9 m/km	5.15 s	88.2 %

In all the tables 5.3, 5.4, 5.5, 5.6, 5.7 and 5.8, two approaches (constant tolerance value and adaptive tolerance value) are compared in terms of maximum vector displacement after simplification, processing time and achieved compression ratio. As it was expected, in both approaches, compression ratio increases by increasing user input value (tolerance value or maximum valid vector displacement). The processing time, however, decreases with more simplification. Because Douglas-Peucker algorithm is a selective approach, where two end points of the original line are selected to be the two end points of the simplified line and then the other points are added to the simplified line based on the tolerance value. In adaptive approach, larger data sets need more time for pre-analysis phase. As it was mentioned before, user cannot predict the maximum vector displacement after simplification in case of using constant tolerance value. However, in case of adaptive tolerance value, the user limits simplification to a certain level by which a specific level of positional accuracy is maintained. All the results show that the adaptive approach can achieve almost the same amount of compression ratio as in the other approach, however, with much lower value of maximum vector displacement.

Vector displacement is used in this research as a positional accuracy measure; however, it can be replaced by any other measure (such as area displacement, percentage of length change, percentage of angularity changes, percentage of curvilinearity changes or any other positional accuracy measure) depending on the user application.

5.1 Summary

In most line simplification algorithms, the user is asked to provide a tolerance value to determine the amount of simplification to be occurred. Usually, the user provides different tolerance values and inspects the quality of simplification visually then selects the most appropriate one. However, the problem is that all lines in the data set are simplified with the same tolerance value and therefore different amount of positional errors are introduced for different lines. Further, the user is, usually, unaware of the effect of simplification on the positional accuracy of the data set.

A new approach (adaptive tolerance value) was introduced in this chapter to solve the problem of existing approaches. In this approach user specifies the acceptable level of positional error and a program (developed by the author) determines a proper tolerance value for each line based on this level. This approach includes a pre-analysis phase. The tests results show that the adaptive approach can achieve almost the same amount of compression ratio as in the case of using constant tolerance value, however, with much less positional errors.

Chapter 6

LINE SEGMENTATION AND MULTI-CRITERIA APPROACH FOR LINE SIMPLIFICATION

6.1 Introduction

Using the same tolerance value in simplifying linear features with different geometric characteristics introduces different amounts of positional errors for different linear features. Therefore, simplifying a data set with the same tolerance value is not a realistic assumption especially when the quality of data is an important issue. In the previous chapter, a new approach was introduced in which users specify a maximum valid positional error instead of tolerance value. Then the proper tolerance value is automatically estimated for each line in the data set. The proper tolerance value for each line is a value by which maximum simplification is achieved while maintaining the user specified positional accuracy. This approach, as it was discussed in the previous chapter, involves a pre-computational phase.

The geometric characteristics of linear features not only vary within a data set from one line to another, but also it varies along a single line. Simplifying whole parts of a linear feature with one tolerance value introduces different amount of positional errors in different parts of the line, when the shape of the line changes along its extent. Therefore it makes sense to

split each linear feature to homogenous segments and simplify each segment with a proper tolerance value. Segmentation of linear features and simplifying the segments using the new approach introduced in the previous chapter is discussed in more details in this chapter.

In the approach introduced in previous chapter, users can only define the maximum valid positional errors as input value for simplification. Due to the variety of GIS applications, other geometric characteristics of linear features might be important in some other applications as well. For example preserving the length of linear features might be important in some applications while preserving angularity of linear features might be important in some other applications. In this chapter, the new approach will be extended in such a way that users can define other inputs to preserve geometric characteristics of linear features to a desired degree in the simplification process.

The test results of applying line simplification on geospatial vector data set in the previous chapter showed that line simplification has significant effect on reducing the file size of vector map files. As linear features lose some of their points during the simplification, this process should be classified as lossy compression. At the end of this chapter, the effect of code compression as a lossless compression on geospatial vector data set will be investigated and the effects of line simplification and code compression will be compared.

6.2 Line Segmentation

A line may be called inhomogeneous when it contains portions that seem different from geometric complexity point of view. Simplifying such a line with a single tolerance value will not result in a uniform simplification and the amount of positional error in different portions may be quite different. Segmentation of lines to more homogenous sections and simplifying each segment with the most suited tolerance value minimizes the positional errors along a line during the simplification.

Some experts in the field have proposed different approaches to segment linear features and find the proper tolerance value for simplification of each segment (Buttenfield, 1991; Plazanet, 1997; McMaster, 1993). They use some measurements of line geometry to segment each line and then they try to classify segments and allocate proper tolerance value to simplify each class of segments.

In this research, a line segmentation approach close to Plazanet (1997) approach with some differences is used. Plazanet (1997) uses inflection points for segmentation of the linear features. Plazanet defines homogeneity based on the variation of the distances between consecutive inflection points. Inflection points divide lines to curvilinear segments, which are the portions of a line in which all angles are in the same direction, either positive or negative. In this approach, first the inflection points are detected. Then the mean (M) of distances $d(IP_i, IP_{i+1})$ along the line is calculated. For each distance $d(IP_i, IP_{i+1})$, the deviation distance D_i is calculated using the following formula:

$$D_i = d(IP_i, IP_{i+1}) - M \qquad\qquad 6.1$$

The inflection points IP_i define the potential location for the segmentation of lines, if $S_i \neq S_{i-1}$, where S_i is the sign of D_i . Based on this approach, the line in figure 6.1 is broken to two segments. The arrow shows the segmentation point.
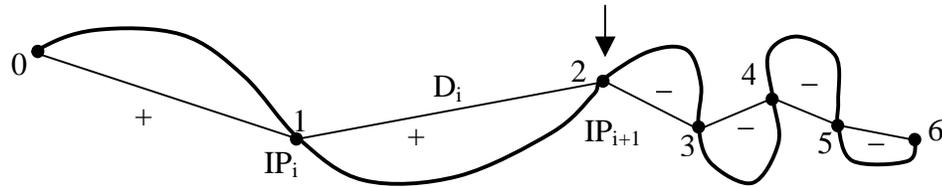

Figure 6.1 Line segmentation

As described above, the criterion for segmentation is the sign of $Di = d(IP_i,\ IP_{i+1}) - M$. This is a very strict criterion; because if $d(IP_i,\ IP_{i+1})$ is slightly larger or smaller than M, a new segment will be created causing several small segments to be created. Therefore, to make segments larger, the following new deviation distances are introduced:

$$D_i = d(IP_i,\ IP_{i+1}) - (M + \sigma) \qquad\qquad 6.2$$

$$D_i = d(IP_i,\ IP_{i+1}) - (M + 2\sigma) \qquad\qquad 6.3$$

$$D_i = d(IP_i,\ IP_{i+1}) - (M + 3\sigma) \qquad\qquad 6.4$$

where σ is standard deviation of $d(IP_i,\ IP_{i+1})$ distances. The standard deviation is a statistic that tells us, in this case, how tightly values of distances are clustered around the mean in the set of distances. Using equation 6.4 for segmentation causes less number of segments and therefore longer segments in a line with respect to equations 6.3, 6.2 and 6.1.

After segmentation of linear features, the same approach introduced in chapter 5 is used for simplification of segments. The first step is pre-calculation phase in which the vector displacements are calculated for each segment of lines using different tolerance values. The set of tolerance values can be defined in the same way as discussed in the section 5.3.1; however, in this case the set of tolerance values are defined for each segment rather than for each line. T_{max} , in this case, is the value that can simplify a segment to a straight line connecting the start point and the end point of that segment. The output of the pre-calculation step is a text file. Figure 6.2 shows the structure of this text file.

As it can be seen from Table 6.1, the text file starts with feature identifier number for the first feature and then continues with part number followed by Seg_start and Seg_end for each segment of the line. Seg_start is the index of the first point of the segment in the original line. Seg_end is the index of the last point of the segment in the original line. For example in the case of figure 6.1 segmentation, Seg_start = 0 and Seg_end = 2 for the first segment and Seg_start = 2 and Seg_end = 6 for the second segment. The reason to have Seg_start and Seg_end in the text file resulting from the pre-calculation phase is to provide the start and end points of segments for the simplification phase. In fact the original data doesn't change in the pre-calculation phase, however, the indices of start and end points of segments in the original lines are recorded in the above-mentioned text file. These indices are then used in simplification phase to identify segments.

Table 6.1 The output file structure of the pre-calculation phase in case of line segmentation.

Structure	Example
ID #	1
Part #	1
Seg_start	0
Seg_end	8
0 0	0 0
T_1 VD_1	255.101 20.7483
T_2 VD_2	510.203 41.4608
T_3 VD_3	765.304 58.494
…..	…..
…..	…..
T_{max} VD_{max}	1530.61 111.425
-3	-3
Seg_start	8
Seg_end	15
0 0	0 0
T_1 VD_1	357.382 19.2386
T_2 VD_2	534.777 52.8402
T_3 VD_3	937.771 89.3645
…..	…..
…..	…..
T_{max} VD_{max}	2342.635 214.154
-3	-3
-2	-2
Part #	2
Seg_start	0
Seg_end	12
0 0	0 0
T_1 VD_1	259.443 17.7157
T_2 VD_2	518.887 62.4915
T_3 VD_3	1037.770 91.8747
…..	…..
…..	…..
T_{max} VD_{max}	2594.430 304.112
-3	-3
-2	-2
-1	-1
ID #	2
……	……

After the pre-calculation phase, the user specifies a maximum valid vector displacement. Then the proper tolerance value for each segment is automatically selected using the text file created in the pre-calculation phase and the segment is simplified using that value. The proper tolerance value for each segment is the maximum tolerance value that has an associated positional displacement less than the user specified level of positional displacement.

6.2.1 Test Results and Evaluation

Simplification of segments rather than lines has been tested using some of the data sets listed in Table 3.2 and Table 5.2. As mentioned before any of the equations 6.1, 6.2, 6.3 and 6.4 can be used for segmentation but the results will be different. Figure 6.2 illustrates simplification of a line in the data set#5 with segmentation using equations 6.1, 6.2, 6.3 and 6.4. As it can be seen in Figure 6.2, the least simplification is achieved when segmentation is done using equation 6.1 and the most simplification is achieved when segmentation is done using equation 6.4. The reason is that the number of segments is more and the segments are shorter when using equation 6.1.

The results of simplification with segmentation on three data sets each with two different user input values are listed in the Tables 6.2, 6.3 and 6.4. These tables present the user input maximum valid vector displacement, equation used for segmentation, maximum vector displacement after simplification, time needed for pre-calculation phase plus time needed for simplification phase and achieved compression ratio, respectively.
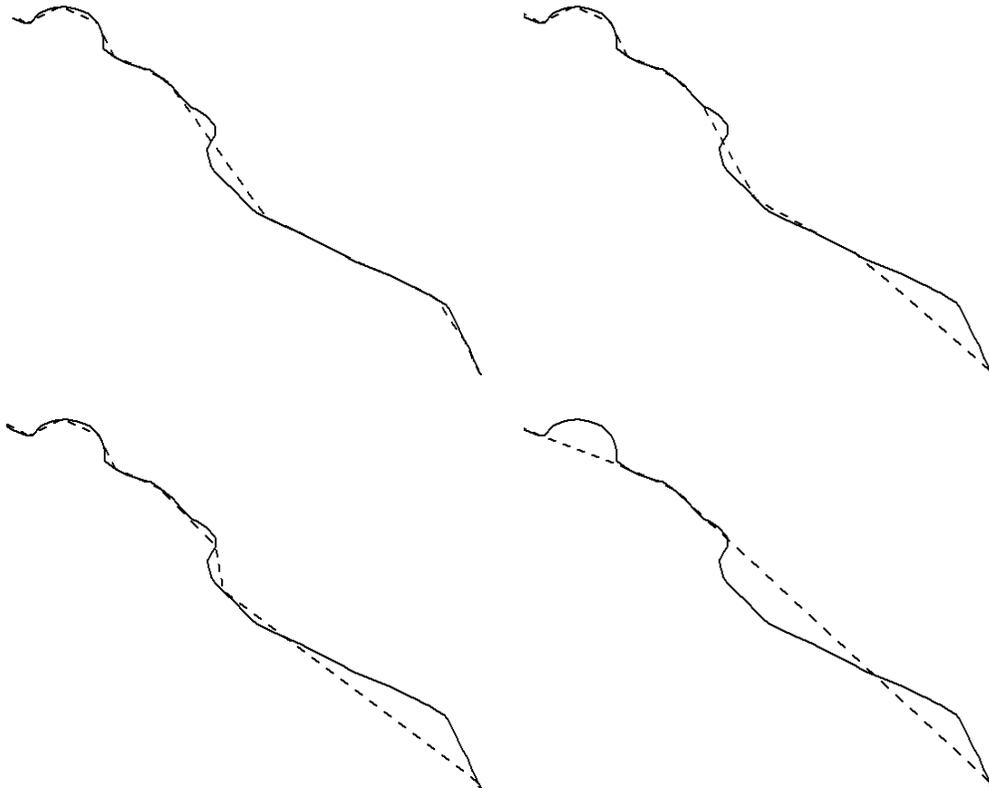
Figure 6.2 Line simplification with segmentation on data set #5 (using vd=2600m/km as user input). Top-left: using equation 6.1; Top-right: using equation 6.2; Down-left: using equation 6.3; Down-right: using equation 6.4.

Table 6.2 Test results of line simplification after segmentation of data set#1

User Input	Equation	Max. Vec. Disp.	Time	Comp. Ratio
VD= 130 m/km	Equation (6.1)	129.9 m/km	8.2 s + 0.51 s	63.6 %
VD= 130 m/km	Equation (6.2)	129.9 m/km	7.2 s + 0.47 s	70.7 %
VD= 130 m/km	Equation (6.3)	129.9 m/km	6.5 s + 0.40 s	75.0 %
VD= 130 m/km	Equation (6.4)	129.9 m/km	5.9 s + 0.36 s	77.9 %
VD=4500m/km	Equation (6.1)	2442.4 m/km	8.2 s + 0.47 s	73.6 %
VD=4500m/km	Equation (6.2)	3904.4 m/km	7.2 s + 0.43 s	82.1 %
VD=4500m/km	Equation (6.3)	4037.86 m/km	6.5 s + 0.35 s	87.9 %
VD=4500m/km	Equation (6.4)	4492.99 m/km	5.9 s + 0.31 s	90.7 %

Table 6.3 Test results of line simplification after segmentation on data set#3

User Input	Equation	Max. Vec. Disp.	Time	Comp. Ratio
VD= 190 m/km	Equation (6.1)	189.82 m/km	11.4 s +0.91 s	56.6 %
VD= 190 m/km	Equation (6.2)	189.82 m/km	10.9 s + 0.78 s	60.6 %
VD= 190 m/km	Equation (6.3)	189.51 m/km	10.3 s + 0.76 s	64.3 %
VD= 190 m/km	Equation (6.4)	189.51 m/km	10.2 s + 0.75 s	65.0 %
VD= 4300 m/km	Equation (6.1)	2475.02 m/km	11.4 s + 0.79 s	62.3 %
VD= 4300 m/km	Equation (6.2)	3341.86 m/km	10.9 s + 0.76 s	66.7 %
VD= 4300 m/km	Equation (6.3)	4229.16 m/km	10.3 s + 0.75 s	70.7 %
VD= 4300 m/km	Equation (6.4)	4229.16 m/km	10.2 s + 0.72 s	71.0 %

Table 6.4 Test results of line simplification after segmentation on data set#5

User Input	Equation	Max. Vec. Disp.	Time	Comp. Ratio
VD= 200 m/km	Equation (6.1)	199.99 m/km	175.6 s +11.0 s	64.7 %
VD= 200 m/km	Equation (6.2)	199.99 m/km	162.1 s+10.32s	70.3 %
VD= 200 m/km	Equation (6.3)	199.99 m/km	145.5 s +9.25 s	75.1 %
VD= 200 m/km	Equation (6.4)	199.99 m/km	135.8 s + 9.13s	76.4 %
VD= 2600m/km	Equation (6.1)	2598.01 m/km	175.6 s+10.73s	69.5 %
VD= 2600m/km	Equation (6.2)	2598.15 m/km	162.1 s+10.26s	75.7 %
VD= 2600m/km	Equation (6.3)	2598.15 m/km	145.5 s +8.79 s	81.2 %
VD= 2600m/km	Equation (6.4)	2598.16 m/km	135.8 s + 8.22s	82.6 %

To compare the results of simplification with segmentation to the results of simplification without segmentation, the results presented in tables 6.2, 6.3 and 6.4 are compared to the results presented in tables 5.3, 5.5 and 5.7. These results clearly show that the compression ratio (or the amount of simplification) is lower in the case of simplification with segmentation. Figures 6.3 and 6.4 show the maximum simplification of the line shown in Figure 6.1 without segmentation and with segmentation, respectively. As it can be seen in Figure 6.3, maximum simplification of the line has resulted in a straight line connecting the start and the end points of the original line, therefore only two vertices remains. However in figure 6.4, maximum simplification has resulted in two straight lines each connecting the start and the end points of one of two segments of the line, therefore, three vertices remains.
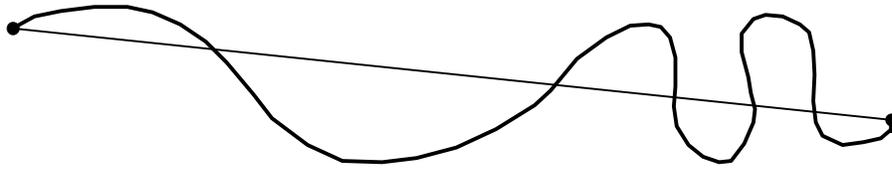
**Figure 6.3 Maximum simplification (without segmentation)**
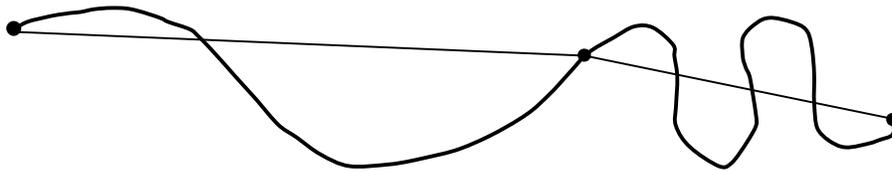
Figure 6.4 Maximum simplification (with segmentation)

The comparison of tables 6.2, 6.3 and 6.4 with tables 5.3, 5.5 and 5.7 also show that simplification with segmentation requires more time for the pre-calculation phase. More time is needed to calculate the sets of tolerance values and associated vector displacements for segments than for lines. Because the number of lines in a data set is less than number of line segments in that data set after segmentation.

All results in tables 6.2, 6.3 and 6.4 show that the compression ratio (i.e. the amount of simplification) increases when we go from equation 6.1 to equation 6.4. The reason is that equation 6.1 creates the shortest segments and more number of segments with respect to the other equations while equation 6.4 creates the longest segments and less number of

segments with respect to the other equations. As the number of segments increases, less number of vertices is removed due to simplification (refer to figures 6.3 and 6.4).

In the tables 6.2, 6.3 and 6.4, it should be noticed that the time needed for pre-calculation phase is the longest when using equation 6.1. The reason is that we have more segments in this case as compared to the cases using the other equations. Therefore, more time is needed to calculate the sets of tolerance values and associated vector displacements for segments to be recorded in the text file (resulting from pre-calculation phase). For the same reason, the time needed for simplification is the longest when using equation 6.1, because there are more segments to be simplified.

6.1 Multi-Criteria Approach

In the new approach defined in chapter 5, the user has control over the vector displacement occurring in the lines during simplification. However, as it was shown in chapter 4, the geometric characteristics of different lines in the data set changes differently when using one single tolerance value to simplify all lines in a data set. In other words, one line may lose 40% of its length while the other line may lose only 5% of its length, when both are simplified using a single tolerance value. What if users want to simplify a data set in such a way that, for example, 75% of length of all lines be preserved after simplification or certain percentage of angularity or density of all lines be preserved in the data set?

Extending the new approach defined in chapter 5 will preserve geometric characteristics to user specified level. The extended approach will be called **multi-criteria approach** (as compared to the **single-criterion approach** used in chapter 5). This approach accepts

multiple criteria from the user as input to the simplification process. In the multi-criteria approach, users can specify three values to preserve three geometric characteristics (length, density and angularity) as well as vector displacement (as positional accuracy criterion). In this approach, "percentage change in line length", "ratio of change in number of coordinates" and "percentage change in angularity" have been used for length, density and angularity measures respectively (refer to section 4.2). The phases in this approach are as follows:

1. The pre-calculation phase: In this phase the positional displacement, percentage change in line length, percentage change in number of coordinates and percentage change in angularity for each line in the data set are calculated using a set of tolerance values for each line. The set of tolerance values for each line starts from zero and ends by the maximum possible tolerance value for that line. The set of tolerance values of lines and corresponding measurements (measurement of vector displacement and three measurements of geometric characteristics) along with the feature identifiers are recorded in four separate text files.

2. User input phase: After pre-calculation, the user specifies one, two, three or all four criteria for simplification. The user has to specify the value of at least one criterion for simplification process and can enter –1 as input for the other criteria to make the program ignore them.

3. Simplification phase: In this phase the program finds proper tolerance values for each line from the text files created in pre-calculation step. The final tolerance value for each line is the minimum value of all tolerance values found for that

line. The reason to select the minimum proper tolerance value for the line is that the minimum value satisfies all user specified criteria. Then each line is simplified using the specified tolerance value.

Multi-criteria approach provides the user with the choice of preserving positional accuracy and geometric characteristics to the desired levels. Using the multi-criteria approach, vector displacements, change in density, length and angularity are all under user's control. This is very beneficial to the user. However, as in this approach, four text files are created in pre-calculation step, the time needed for pre-calculation is higher than the single-criterion approach. Simplification steps will also need more time in multi-criteria approach with respect to the single-criterion approach, because the proper tolerance values for each line need to be found from more than one text file.

6.3.1 Test Results and Evaluation

The multi-criteria approach has been tested using the different data sets listed in Table 3.2 and Table 5.2. As expected, the pre-calculation phase takes more time in the multi-criteria approach as compared to the single-criterion approach. However, the results of using the multi-criteria approach regarding compression ratio and time totally depend on the user specified criteria. Figure 6.5 illustrates the results of two simplifications on a line in data set 5. The more straight simplified line (a) is the result of simplification when the user specifies to preserve at least 30% of length of lines and the more angular simplified line (b) is the result of simplification when the user specifies to preserve at least 30% of angularity of lines. The figure clearly show that, in the first simplification (a), the angularity, has not

been preserved as well as vector displacement and density while in the second simplification, the angularity of simplified line is very close to the angularity of original line.
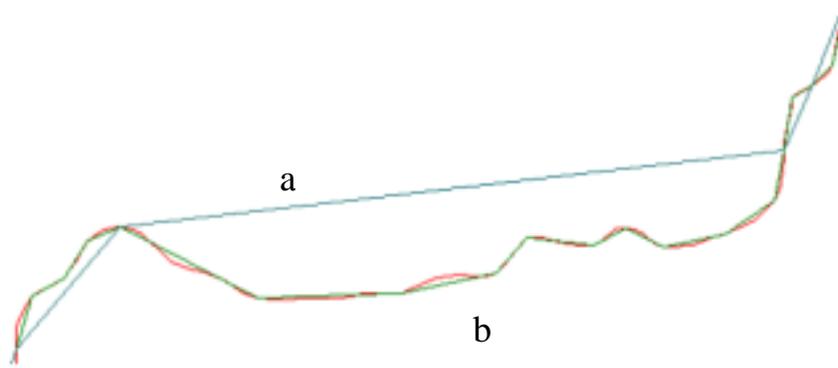

Figure 6.5 Two simplification of a line: (a) Preserving at least 30% of length, (b) preserving at least 30% of angularity

Table 6.5 lists the result of simplification of data set 5 using the multi-criteria approach. In this table:

- o The first column of the table shows the amounts that the user has specified for length criterion;

- o The second column shows these amounts for angularity criterion;

- o Third column shows the maximum vector displacement after simplification;

- o Fourth column shows the time used for simplification;

- o Fifth column includes compression ratios achieved by simplification.

Table 6.5 Test results of multi-criteria approach on data set 5

Length	Angularity	Max. Vec. Disp.	Time	Comp. Ratio
30 %	-1	20559.9 m/km	8.50 s	83.0 %
-1	30 %	3519.9 m/km	9.03 s	71.3 %
10 %	20 %	5172.8 m/km	12.2 s	73.5 %
20 %	10 %	5591.2 m/km	12.0 s	75.1 %

The pre-calculation time is the same for all cases listed in table 6.5. The table clearly indicates that preserving more percentage of geometric characteristics and less displacement of lines results in less simplification and therefore lower compression ratio.

6.2 Code Compression

The test results in this research has shown that line simplification of linear features in geospatial vector data set has significant effect on reducing the size of files and therefore, it can be considered as a vector data compression technique. As linear features lose some of their points during the simplification, this process should be classified as lossy compression. Line simplification is in fact a content-based compression technique because the content of maps (linear features) are analyzed and simplified during this process. To maximize the compression ratio, code compression has been integrated with line simplification in this research. Huffman code compression technique has been used to compress the data. The reason to choose this technique is that it is one of the most commonly used code compression techniques and, more important reason, is that it is a

lossless compression technique in which no data is lost during the compression. For vector geospatial data we can not use a lossy code compression technique because we need the data (the coordinates of points and identifier numbers) to be completely restored. The data needs to be decompressed to be usable by the user. As the time and compression ratio are two important factors in compressing vector data for Internet-based GIS applications, the effect of integrating code compression (compressing and decompressing data) and line simplification on these two factors will be investigated.

Table 6.6 illustrates the result of integrating code compression with line simplification on data set 1, which is a small scale data set and data set 5, which is a large scale data set. Two different approaches, namely, constant tolerance value and adaptive tolerance value have been used for line simplification process. The first column of this table lists the data set used for each test and the user input value. The second column lists the time needed for simplification process (equation 6.4 has been used for segmentation). The third column shows the achieved compression ratio with only simplification process. The fourth column shows the time needed for code compression and decompression using Huffman code compression technique and the last column shows the compression ratio when code compression is integrated with line simplification.

Table 6.6 Test results of Integrating line simplification and code compression

Data set	Simplification Time	Comp. Ratio	Huffman Time	Final Comp. Ratio
Data set 1 T= 1000m	0.17 s	79.3 %	0.78 s + 0.42 s	80.7 %
Data set 1 VD = 130 m/km	0.36 s	77.9 %	0.84 s + 0.43 s	79.3 %
Data set 5 T = 100 m	4.26 s	82.8 %	14.64 s + 8.16	85.36 %
Data set 5 VD = 2600 m/km	8.22 s	82.6 %	14.86 s + 8.56 s	85.11 %

Table 6.6 clearly shows that the simplification time of data set 5 is much more than the simplification time of data set 1. The reason is that data set 5 is larger in scale than the data set 1 and, therefore, it includes much more number of lines than data set 1. It can also be seen from the table that the time needed for compression and decompression of the data is much more than the simplification time. It should also be noted that compressing the data is done at the server side and decompressing data is done at the client side (i.e. in an Internet-based GIS applications). Therefore, both times are important factors in data transmission. The more important point in table 6.6 is that we cannot get much more compression ratio when we integrate code compression with line simplification. In case of data set 1 the effect

of code compression on the compression ratio is less than 2% and in the case of data set 5 it is less than 4% when comparing the third column and the fifth column of the table. These results show that code compression (at least the Huffman technique) is not very effective in compressing vector geodata sets while line simplification significantly reduces the file size of vector data sets.

6.3 Conclusion

Linear features are not usually homogenous lines from geometry point of view because the geometric characteristics of linear features usually vary along the lines. Therefore simplifying the whole parts of a linear feature with one tolerance value introduces different amount of positional errors in different parts of the line. Therefore it makes sense to split each linear feature to homogenous segments and simplify each segment with a proper tolerance value. Segmentation of linear features and simplifying each segment with its proper tolerance value results in better simplification regarding positional errors. However, the results showed that the time needed for pre-calculation as well as simplification is more when segments are simplified rather than entire lines. The compression ratio was also lower in the case of segmentation.

Preserving geometric characteristics of linear features to a certain level can be as important as reducing vector displacements in simplification process. The single-criterion approach was extended to multi-criteria approach by which the percentage change in length, percentage change in density and percentage change in angularity as well as vector displacement are under user's control. All user specified criteria are fulfilled with this

approach and the time needed for pre-calculation and simplification and compression ratio depend on user specified number of criteria (one to four) and the values of these inputs.

Finally, to maximize the compression ratio, code compression has been integrated with line simplification process. However, the results showed that code compression and decompression needs much more time with respect to line simplification and in the other hand the effect of code compression on compression ratio is not comparable to the effect of simplification on compression ratio. As the time and compression ratio are two important factors in data transmission, code compression doesn't seem to be an ideal compression technique for compressing geospatial vector data sets.

Chapter 7

CONCLUSIONS AND RECOMMENDATIONS

7.1   Summary

The Internet has already become one of the major sources of geospatial information and offers different GIS functionalities. Nowadays, many Internet-based GISs provide users with the most updated geodata and on-line geoprocessing services. Internet-based GISs are based on client-server architecture. One of the main challenging issues in Internet-based GIS is the speed of data transmission through Internet. Although improving networking technology may increase the speed of transmission, however, this issue still remains a challenge. The other solution to this problem is data compression by which the size of files is reduced.

Geospatial data is categorized as raster and vector data. Internet transmissions of raster and vector data are two different issues, because these two data models have completely different structures. Efficient transmission of raster data through Internet has already been very successful due to the nature of raster data model (Buttenfield, 1999). However, Internet transmission of vector geospatial data still remains a challenge. Many of GIS

analysis functions use vector data. Vector geospatial data sets have usually huge volumes and file sizes increase by increasing geometric complexity of features in the file.

The research has investigated the efficiency of line simplification technique for vector data compression. Two important factors in Internet-based GISs, namely, compression ratio and the time needed for simplification have been considered in this investigation. The effect of line simplification on positional displacement and geometry of linear features has also been investigated. Due to the lack of user's control on these effects in the existing simplification techniques, a new approach had to be introduced for simplification of linear features (eg. roads and rivers).

The development of a new approach for simplification of linear vector data for Internet_based GIS applications, to compress data while preserving the positional accuracy and geometry of linear features to the user-specified level, was the major objective of this research. This objective was successfully achieved. In this approach, the user has complete control on positional displacement and on geometry of linear features in the simplification process of vector geospatial data sets. The newly proposed approach has been called **multi-criteria approach** in which the user specifies one to four criteria based on his/her application. Thus each line is simplified with its most suited tolerance value. All the user specified criteria are fulfilled in the newly proposed approach. Simplification of linear features with segmentation of lines to homogenous sections has also been investigated in this research. Finally, code compression has been integrated with line simplification to maximize the compression ratio. The efficiency of this integration on compressing vector

geospatial data sets from time and compression ratio point of view has also been investigated.

7.2   Conclusions

The following conclusions address the main findings of this research:

- Line simplification (Using Douglas-Peucker algorithm) has significant effect on reducing the file size of vector geospatial data sets, therefore it can be considered as a suitable compression technique for compressing vector maps in Internet-based GISs. However it is a lossy compression, because linear features lose some of their points during simplification process. The compression ratio depends on the amount of simplification. Using Douglas-Peucker algorithm, more simplification requires less compression time and results in higher compression ratio.

- Line simplification introduces positional errors in the data set by creating a positional discrepancy between the original line and simplified line. The amount of discrepancy depends on the level of the desired simplification, which is determined by the user specified tolerance value in current simplification algorithms. Test results in this research showed that the amount of positional discrepancy also depends on the geometric characteristics of lines. Using one tolerance value to simplify all lines in a data set results in different amount of positional discrepancy for different lines. This is mainly because linear features in geospatial vector data sets usually have different geometry.

- Test results showed that simplifying a vector data set with constant tolerance value results in different amounts of changes in geometric characteristics of different lines. In other words, the percentage change in geometric characteristics (e.g. length, angularity and density) after simplification of lines with a constant tolerance value are completely different for different lines.

- The new proposed approach (adaptive tolerance values or single criterion approach) introduced in the research by which the vector displacements of lines in the data set are under the user's control. The user specifies maximum valid vector displacement rather than tolerance value for simplifying a data set. Then the program calculates the proper tolerance value for each line using the results of a pre-calculation phase. Then each line is simplified using its proper tolerance value. The results showed that in the adaptive approach we can achieve almost the same amount of compression ratio as the case with the constant tolerance approach, however with much less vector displacements. The time needed for pre-calculation phase doesn't make any problem in Internet-based GISs, because it is done only once (for a certain data set) and the results is saved at the server side. The time needed for the simplification phase in the new approach is a bit higher than in the traditional approach (constant tolerance value). This is mainly because of the need to find the proper tolerance value for each line based on the user specified tolerance value. The test results also showed that the vector displacement after simplification is always equal or less than the user specified value.

- The single criterion approach was extended to multi-criterion approach by which not only the vector displacements are under user's control, but also percentage changes in geometric characteristics of lines (length, density and angularity). In this approach, the user can specify one to four criteria for simplification based on his/her application and the program calculates the most suitable tolerance value for each line. The time needed for the pre-calculation phase in the multi-criteria approach is higher than in single-criterion approach due to the creation of four text files in this approach. The compression ratio and the time needed for simplification in the multi-criteria approach depend on the number of user specified criteria and the amount of these values.

- Segmentation of lines to more homogenous segments results in better simplification from vector displacement and geometric characteristics point of view. However, the results showed that the time needed for pre-calculation and simplification is more when segments are simplified rather than entire lines.

- Code compression (Huffman coding) was integrated with line simplification to maximize the compression ratio. However, the results showed that the effect of code compression on compression ratio is much lower than the effect of line simplification. In the other hand, code compression and decompression needs much more time than line simplification. As the time and compression ratio are two important factors in data transmission, code compression doesn't seem to be an ideal compression technique for compressing geospatial vector data sets.

7.3   Recommendations

The following recommendations are made for further work on this research:

- Each line simplification algorithm has its own way to simplify the lines, therefore, the effect of simplification on positional accuracy and geometric changes differ from one algorithm to the other algorithm. Douglas-Peucker algorithm has been used in this research because it is a global routine and the most common used line simplification algorithm. It is recommended that other simplification algorithms be investigated and compared to Douglas-Peucker algorithm from this research point of view.

- In this research, McMaster's (1986) defined measures have been used to measure the changes in geometric characteristics of lines. It would be worthwhile to use other measures defined by other researchers and investigate the effect of simplification on those measurements.

- Vector displacement has been used as a criterion for positional accuracy of data set after simplification; however, other measures such as polygonal displacement can also be used or even integrated with vector displacement.

- Plazanet (1997) with some modification has been used for segmentation of lines. It should be worthwhile to investigate other segmentation approaches from time and compression ratio point of view.

- Huffman coding has been used as a code compression technique in this research, however, other lossless code compression techniques may also be used and the results may be compared to the results achieved by Huffamn coding.

- In this research, linear features (such as roads and rivers) have been simplified. This work can be extended to include the simplification of polygons such as contour lines.

- Preserving topology is an important issue in GIS. Line simplification may result in some conflicts in topological relationships of features. For example a line that doesn't intersect itself in original data set may intersect itself after simplification or two lines that don't intersect each other may intersect each other after simplification. Preserving topology is a very complicated issue and needs to be investigated in depth.

These problems will provide challenges for years to come. Solving these problems will be an enormous extension to the efficiency of Internet-based GIS applications.

REFERENCES

Buttenfield, Barbara P., 1999. "Progressive Transmission of Vector Data on the Internet: A Cartographic Solution". Proceedings, 18th International Cartographic Conference. Ottawa, Canada. pp. 16-19. August 1999.

Buttenfield, Barbara P., 1991. "A Rule for Describing Line Feature Geometry". In "Map Generalization". Chapter3. pp. 150-171. Buttenfield and McMaster Ed. Longman Scientific & Technical. London.

Buttenfield, Barbara P., 1984. "Line Structure in Graphic and Geographic Space". Unpublished Ph.D. Dissertation, University of Washington.

Campos, Arturo San Emeterio, 2000. "The introduction to Data Compression".

http://www.arturocampos.com/cp_ch1.html

Deveau, Terry J., 1985. "Reducing the number of points in a plane curve representation". Proceedings of Seventh International Symposium on Computer-Assisted Cartography, AutoCarto 7. Falls Church, VA.: American Society of Photogrammetry and American Congress on Surveying and Mapping, pp. 152-160. In: Buttenfield, Barbara P., 1991. "A Rule for Describing Line Feature Geometry". In "Map Generalization". Chapter3. pp. 150-171. Buttenfield and McMaster Ed. Longman Scientific & Technical. London.

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

Dutton, G. H., 1999. "Introduction and Problem statement", In "A hierarchical coordinate system for geoprocessing and cartography". Lecture notes in Earth Science 79. Berlin: Springer-Verlag.

Goebel, Greg, 2002. "Introduction / Lossless Data Compression"
http://www.vectorsite.net/ttdcmp1.html

Ioup, Juliette W., 2000. "Vector Map Data Compression with Wavelets". The Royal Institute of Navigation. Journal of Navigation. Vol. 53. No.3.

Ioup, Juliette W., 2000. "Vector Map Data Compression with Wavelets". The Royal Institute of Navigation. Journal of Navigation. Vol. 53. No.3.

Jasinski, M. J., 1990. "The Compression of Complexity Measures for Cartographic Lines". NCGIA Technical report 90-1. National Center for Geographic Information & Analysis, Department of Geography. State University of New York at Buffalo, Buffalo, New York 14260.

Jenks, G. F., 1989. "Geographic Logic in Line Generalization". Cartographica, 26, pp. 27-41. In: Veregin, H., 2000. "Quantifying positional error induced by line simplification". International Journal of Geographical Information Science, Vol. 14, No. 2, pp. 113-130.

Ladino, Jeffrey N., 1996. "Data Compression Algorithms". http://www.ccs.neu.edu/groups/honors-program/freshsem/19951996/jnl22/jeff.html

Lelewer, D.A. and Hirschberg, D.S., 1987. "Data Compression". Computing Surveys 19,3 (1987), 261-297. Reprinted in Japanese BIT Special issue in Computer Science (1989), 165-195.

Little, A. R., 1989. "An Evaluation of Selected Computer-Assisted Line Simplification Algorithms in the Context of Map Accuracy Standards. Technical Papers, ASPRS/ACSM Annual Convention (Falls Church: American Society for Photogrammetry and Remote Sensing and American Congress on Surveying and Mapping), 5, pp.122-132. In: Veregin, H., 2000. "Quantifying positional error induced by line simplification". International Journal of Geographical Information Science, Vol. 14, No. 2, pp. 113-130.

McMaster, R. B., 1993. "Knowledge Acquisition for Cartographic Generalization: Experimental Methods". In: Muller, J-C, Lagrange, J. P. and Weibel, R. (eds). "GIS and Generalization: Methodological and Practical Issues". London: Taylor & Francis, pp. 161-79.

McMaster, R. B., 1989. "The integration of simplification and smoothing algorithms in line generalization". Cartographica, 26 p. 101-121.

McMaster, R. B., 1987a. "Automatic Line Generalization". Cartographica, 24(2), pp. 74-111.

McMaster, R. B., 1987b. "The Geometric Properties of Numerical Generalization". Geographical Analysis, 19(4), pp. 330-346.

McMaster, R. B., 1986. "A Statistical Analysis of Mathematical Measures of Linear Simplification". The American Cartographer, 13(2), pp. 103-116.

Plazanet, C., 1997. "Modeling Geometry for Linear Feature Generalization", In book ESF/NSF 1995 Summer Institute in Geographic Information Research: Bridging the Atlantic, in Portland Maine (USA). Ed. Taylor & Francis 1997, Part 3, Chapter 17, pp. 264-279.

Plazanet, C., Affholder, J-G., Fritch, E., 1996. "The Importance of Geometric Modeling in Linear Feature Generalization". Cartography and Geographic Information Systems, 22(4), pp. 291-305.

Plewe, Brandon, 1997. "GIS Online: Information Retrieval, Mapping, and the Internet". Santa Fe, NM: Onword Press.

Saju, Vimil, 2002. "The Huffman Compression Algorithm". http://www.howtodothings.com/showarticle.asp?article=313

Shahriari, Nadia and Tao, C. Vincent, 2002. "Minimizing Positional Errors in Line Simplification Using Adaptive Tolerance Values". Spatial Data Handling, pp. 153-166, ISBN 3-540-43802-5, Springer.

Tobler, Waldo, 1966. "Numerical Map Generalization". In John Nystuen, ed. Michigan Interuniversity Community of Mathematical Geographers, Discussion paper 8, Ann Arbor: University of Michigan, p. 25. In: Buttenfield, Barbara P., 1991. "A Rule for Describing Line Feature Geometry". In "Map Generalization". Chapter3. pp. 150-171. Buttenfield and McMaster Ed. Longman Scientific & Technical. London.

Varga, Margaret, 1998. "Content-based Image Compression". Multi-media data compression, BMVA technical meeting on December 2nd , 1998 at the British Institute of radiology, 36 Portland Place, London.

Vemuri, B., Sahni, S., Chen, F., Kapoor, C., Leonard, C., and Fitzsimmons, J., 2002. "State of the Art Lossless Image Compression. Encyclopedia of Optical Engineering, Ed. Ronald Driggers and Ellen Lichtenstein, Marcel Dekker Inc., 2002.

Veregin, H., 2000. "Quantifying positional error induced by line simplification". International Journal of Geographical Information Science, Vol. 14, No. 2, pp. 113-130.

Veregin, H. and Dai, X., 1999. "Minimizing Positional Error Induced by Line Simplification". Proceedings of the International Symposium on Spatial Data Quality 1999. The HongKong Polytechnic University, HongKong.

Weise, Ulrike, 2001. "Internet GIS".

http://www.geog.ubc.ca/courses/klink/g516/talks_2001/Internet%20GIS.htm

White, Ellen R., 1983. "Perceptual Evaluation of Line Generalization Algorithms". Unpublished Master's Thesis, University of Oklahoma.

APPENDIX A

DESCRIPTION OF THE DEVELOPED SOFTWARE

Several C++ programs were developed in implementation phase of this research. These programs are run in DOS command prompt. The input data format of these programs is ESRI Shape format. ESRI Shape format is a vector data format. Each shape file may consist of points, lines or polygons. Shape files consisting of linear features (roads and rivers) were used as input to the programs developed in this research. User can specify four criteria for simplification based on this implementation. However, other criteria can also be programmed to provide users with more options for simplification.

1. Simplification with Constant Tolerance Value

The program simplifies all the linear features in the data set using the user specified tolerance value and compresses the result using Huffman coding technique.

Usage: advgis *<shape file name>*

Input: Shape file name

                  Single tolerance value (T)

Output: If T=0, the outputs are:

                  Maximum possible value for T

                  Maximum possible vector displacement after simplification

             If T<>0, the outputs are:

Simplified data set in binary format

Compressed simplified data set in binary format

Maximum vector displacement after simplification

The time spent for code compression

The total time spent for simplification and code compression

2. Simplification with Adaptive Tolerance Value

Simplification with Adaptive Tolerance Value includes two phases: pre_calculation and simplification.

2.1 Pre_Calculation Phase

The program calculates the amounts of vector displacements for all linear features assuming a set of tolerance value for each linear feature. For more details refer to Table 5.1.

Usage: advgis1 *<shape file name>*

Input: Shape file name

Output: ftvd.txt (refer to Table 5.1)

             Maximum possible value of vector displacement

             Maximum possible value of T

2.2 Simplification Phase

The user specifies maximum valid vector displacement for the data set. The program uses ftvd.txt file from pre_calculation phase to find a proper tolerance value for each linear feature and simplifies each line using the proper tolerance value specified for that line. The result is compressed using Huffman coding technique.

Usage: advgis2 *<shape file name>*

Input: Shape file name

             Maximum valid vector displacement for the data set

             ftvd.txt from pre_calculation phase

Output: Simplified data set in binary format, using adaptive approach

Compressed simplified data set in binary format

Maximum vector displacement after simplification

3. Segmentation and Simplification of Segments with Constant Tolerance Value

The program segments all the linear features in the data set using one of the equations 6.1, 6.2, 6.3, 6.4. Then simplifies all the segments using the user specified tolerance value and compresses the result using Huffman coding technique.

Usage: advgiss <*shape file name*>

Input: Shape file name

Single tolerance value (T)

Output: If T=0, the outputs are:

Maximum possible value for T

Maximum possible vector displacement after simplification

If T<>0, the outputs are:

Simplified data set in binary format

Compressed simplified data set in binary format

Maximum vector displacement after simplification

The time spent for code compression

The total time spent for simplification and code compression

4. Segmentation and Simplification of Segments with Adaptive Tolerance Value

Segmentation and simplification with Adaptive Tolerance Value includes two phases: pre_calculation and simplification.

4.1 Pre_Calculation Phase

The program segments all the linear features in the data set using one of the equations 6.1, 6.2, 6.3, 6.4. Then calculates the amounts of vector displacements for all segments assuming a set of tolerance value for each segment. For more details refer to Table 6.1.

Usage: advgiss1 *<shape file name>*

Input: Shape file name

Output: ftvd.txt (refer to Table 6.1)

                Maximum possible value of vector displacement

                Maximum possible value of T

4.2 Simplification Phase

The user specifies maximum valid vector displacement for the data set. The program uses

ftvd.txt file from pre_calculation phase to find a proper tolerance value for each segment

and simplifies each segment using the proper tolerance value specified for that segment.

The result is compressed using Huffman coding technique.

Usage: advgiss2 *<shape file name>*

Input: Shape file name

                Maximum valid vector displacement for the data set

                ftvd.txt from pre_calculation phase

Output: Simplified data set in binary format using adaptive approach

                Compressed simplified data set in binary format

                Maximum vector displacement after simplification

5. Multi_Criteria Approach

Multi_criteria approach includes two phases: pre_calculation and simplification.

5.1 Pre_Calculation Phase

The program segments all the linear features in the data set using one of the equations 6.1,

6.2, 6.3, 6.4. Then calculates the amounts of vector displacements, percentage change in

density, percentage change in length and percentage change in angularity for all segments

assuming a set of tolerance value for each segment (Refer to 6.3 for more details).

Usage: advgisf1 *<shape file name>*

Input: Shape file name

Output: ftvd.txt, fpp.txt, fratiolen.txt and fratioang.txt

                Maximum possible value of vector displacement
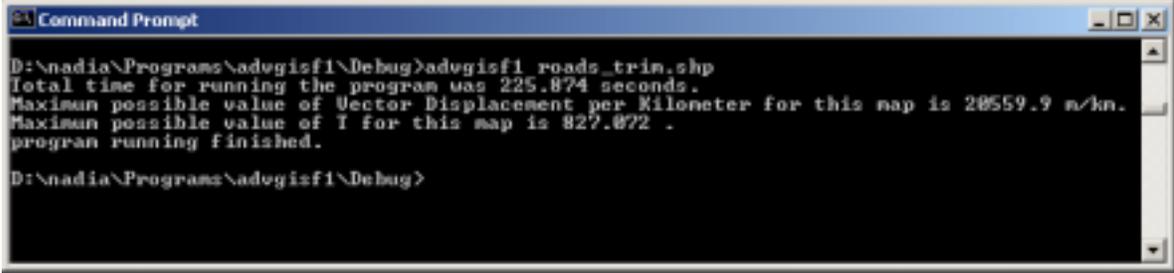
                Maximum possible value of T

Figure A.1 The results of pre-calculation phase on data set 5.

5.2 Simplification Phase

The user specifies one or more criteria for simplification. The program uses ftvd.txt, fpp.txt, fratiolen.txt and fratioang.txt file from pre_calculation phase to find a proper tolerance value for each segment and simplifies each segment using the proper tolerance value specified for that segment. The result is compressed using Huffman coding technique.

Usage: advgisf2   *<shape file name>*.shp

Input: Shape file name

         Maximum valid vector displacement for the data set (or "–1" to ignore this criteria)

         Maximum percentage of density to be preserved (or "–1" to ignore this criteria)

         Maximum percentage of line length to be preserved (or "–1" to ignore this criteria)

         Maximum percentage of angularity to be preserved (or "–1" to ignore this criteria)

         ftvd.txt, fpp.txt, fratiolen.txt and fratioang.txt from pre_calculation phase

Output: Simplified data set in binary format using multi_criteria approach

Compressed simplified data set in binary format

Maximum vector displacement after simplification

Figure A.2 The results of simplification phase on data set 5 with only one criterion.

Figure A.3 The results of simplification phase on data set 5 with two criteria.

Figure A.4 The results of simplification phase on data set 5 with three criteria.

Figure A.5 The results of simplification phase on data set 5 with four criteria.

6. Binary to Generate Conversion

To visualize the result of simplification, the simplified data is converted from Binary format to ESRI Generate format. ESRI Generate format is a text file format, which can be displayed using ESRI Arcview software.

Usage: bindoublegen *<binary file name>*

Input: Binary file name

Output: Simplified data set in ESRI Generate format

7. Data Decompression

The program decompresses a data set, which has been compressed using Huffman coding compression technique. The output of this program can be converted to ESRI Generate format using "bindoublegen" program.

Usage: dechufnew *<compressed file name>*

Input: Compressed file name

Output: Decompressed data set in Binary format