

Generalization of Spatial Data for Web Presentation

Xiaofang Zhou, Department of Computer Science, University of Queensland, Australia, zxf@csee.uq.edu.au

Alex Krumm-Heller, Department of Computer Science, Australian National University, Australia

Volker Gaede, Oracle Consulting Hannover, Germany, vgaede@de.oracle.com

Abstract

In spatial applications, it is often necessary to use spatial objects with varying degree of detail. As spatial objects are typically displayed in a way that allows human users not to discern unnecessary details, it would suffice to draw an abstraction of the spatial data preserving its characteristics. Generalization, the process to derive such a less detailed representation, may lead to a remarkable reduction of the computational overhead involved with displaying complex spatial objects. It can also benefit data transfer from one site to another. With a growing number of interactive spatial applications on the web, spatial data generalization becomes increasingly important. In this paper, we investigate how database systems can support generalization of spatial data. We introduce the concept of visual significance of spatial objects for interactive spatial applications, and discuss efficient ways to produce a good quality generalized map for web-based spatial applications.

Keywords: visual significance, spatial data set generalization, z-ordering

1 Introduction

During the last two decades, spatial database systems have become a vital part of Geographical Information Systems (GIS) for storing and accessing spatial data. Their role is pronounced by the steadily increasing amount of spatial data maintained in such systems. While the data stored in a database represents the finest level of details available, for a given application it is often desirable to use a level of details suitable for the application. For example, when a set of spatial objects are selected from a database in order to draw a map, those objects or parts of objects which are not discernible due to limited display area can be removed. The derivation of an abstract representation is known as *generalization*. Generalization derives from a source dataset a target dataset at a reduced scale whose contents and complexity have been reduced in such a way that the structural characteristics of the source data are maintained for a given application. By removing excessive and non-relevant details, it is possible to derive a representation of the source data that is much more suitable for the given application scenario. In the past, this has been the major motivation for research into generalization, mostly driven by the intention to support cartographers in producing high quality maps [6,3]. Generalization can reduce the data volume considerably since a target data set typically consists of fewer and simpler data objects than the source data. This performance aspect of generalization becomes increasingly important with interactive spatial applications using spatial vector data on the Internet.

A typical architecture for Internet-based spatial applications uses a spatial database to store spatial and non-spatial data. The database can be browsed by means of a Web-browser capable of running Java applets. Generating requests and displaying the result data is handled by the applet that runs on the user's machine. On the server site, a database web server translates the user's request into queries against the spatial database to fetch qualifying objects. These objects are then encoded according to some spatial data transfer protocol (such as [5]), and transferred to the applet over the network. After decoding the data on the client side, they are drawn on the screen. That is, the following major steps are involved: data retrieval (handled by the database system), data encoding (handled by the database server), data transferring over the network, data decoding and drawing (handled by the applet). Generalization can improve the performance on both the database side (eg., less data to encode, smaller amount of data to transfer) and the application side (eg., fewer spatial objects to draw, simpler shapes). Obviously, once the objects are retrieved from the database, they can be simplified (typically before the data-encoding step) on the server site by removing parts of an individual spatial object or finding a suitable

abstraction that preserves its characteristics. We call this type of simplification *object generalization*. Most of previous studies in generalization is focused on object generalization [6]. In this paper, we propose a complementary type of generalization, *data set generalization*. It deals with removing objects from data sets and addresses the issue of spatially significant objects (eg., objects that are visually recognizable when being displayed). It approaches the problem of generalization from a database perspective, treating generalization as an integral part of database query processing. A database query generated from a user's request is modified taking into account of how these data are to be used. If some objects cannot be perceived when being displayed in the web browser, data set generalization tries not to retrieve them from the database at all. The benefit of a smaller amount of data fetched from the database flows on to other steps. However, generalization can degrade the quality of map if the characteristics of the original map are not preserved during generalization. In addition, the data set generalization step may also increase the overall processing costs. Thus, it is crucial to find an efficient data generalization method that can significantly reduce the amount of data, and at the same maintain the key characteristics of the original data set, with as little as possible overhead.

In this paper, we investigate the problem of data set generalization from a database perspective, for the purpose of supporting efficient web-based spatial applications which use maps for navigation. Retrieval of spatial data is a time-consuming operation. Spatial objects are typically large. For example, the boundary of a lake can be represented by thousands of points. It is a waste of resources if some objects are fetched from the database but are found later as insignificant to the application. Based on this observation, we introduce a concept of *visual significance*, and a method to determine the visual significance of spatial objects by examining their index entries rather than the objects themselves. In other words, the full geometry of a qualified object is only retrieved from the database when it is likely to be used by the application. Checking whether an object is significant at the index level can be done efficiently because index entries are much smaller in size and simpler in structure in comparison to the objects they represent.

2 Background

2.1 Generalization of Spatial Objects

Generalization has been the subject of intensive research in the area of cartography and computer science [3]. Besides proposing numerous algorithms for simplifying spatial objects in a way that they bear 'maximum similarity' with the source objects, this research is also characterized by a strong emphasis on correctness aspects [6]. What makes generalization so difficult is that there is no unique solution, but numerous constraints have to be taken into account during the process of generalization. Following [7], the following four constraints have to hold for spatial data that is to be displayed to humans: (1) *Metric constraints*: This class of constraints tries to ensure that all details of spatial data are perceptible for a human observer. Examples of metric constraints are minimal separability, minimal size and minimal width. (2) *Topological constraints*: Existing topological relations between source data objects should be preserved by the generalization process. For example, neighboring spatial objects should also be adjacent after generalization. Examples of topological relations include intersection, adjacency, and containment. (3) *Semantic constraints*: During the process of generalization, the semantics of objects may need to be considered. For example, a river flowing parallel to a road should not bear the name of the road after generalization. (4) *Gestalt constraints*: This type of constraint aims at capturing the overall impression of a scenario and is concerned with perceptual criteria such as maintaining the distribution and the arrangement of features. Noticeably, Gestalt constraints are the most difficult type to enforce and it is very hard to translate them in terms of operations. Furthermore, they can only be checked after all other types of constraints have been enforced.

Clearly, it is not possible to perform satisfactory generalization by focusing on one type of constraint. The process of generalization has to allow for all types of constraints simultaneously. To ensure that all constraints are met after the generalization process, many objects need to be processed at least twice. Typically, the first pass aims at identifying topological relations among the objects. The second pass then applies the generalization operations. A third pass may be necessary to check the constraints. If the constraints are not met, step two and three have to be repeated using a different parameter set.

The generalization process consists of some basic operations [3]. We classify these operations into the following three categories: (1) *Selection*: This operation is to identify a subset of representative objects from a given set for generalization. (2) *Simplification*: This operation addresses the issue of which part of an object should be displayed. Certain conditions, such as preserving the similarity between generalized and source data and maintaining their topological relationship are important for this operation. (3) *Tokenization and amalgamation*: Objects that are important enough to be displayed but too small to be recognized have to be replaced by some visible token. It might be necessary to amalgamate several objects to form a new one; and it is sometimes necessary to displace some objects to preserve their topological relationship. Data set generalization is the first step of generalization. The operations for simplification, tokenization and amalgamation are applied to the objects selected from the database. In this paper, we consider the selection operation to support the metric and Gestalt constraints.

2.2 Visual Significance

The purpose of generalization is to identify for a given data set *significant objects* and *significant parts of the objects*. We distinguish two types of “significance” here. The first one is determined by the semantics of the request. For example, users searching for crop data in a particular region typically have no interest in data irrelevant for their request. Therefore, crop data is considered as significant. For this type of significance, it is possible for the user to *explicitly* specify as attribute constraints, which can be readily used by the underlying database system for object retrieval. This is in contrast to the second type of significance, which deals with *implicit* constraints inferred from the settings of the application and the request. An example of the second type of significance is that for a given application request it is possible to infer constraints that objects smaller than a certain size should not be displayed because they are too small to be discernible.

The conditions to describe the second type of significance depend on the intention of the application and the success of generalization highly depends on the kind and quality of the implicit conditions. In the past, it has been argued that human involvement is necessary to find such conditions [6]. Interactive spatial applications which do not display all the data items meeting the explicit constraints have to derive automatically implicit conditions allowing the identification of “visually significant” objects. Intuitively, we say that a spatial object is visually significant if it is in one of the following cases: (1) *Large objects*: Objects larger than certain size are important and therefore, they should be selected. Here, size refers not only to area, but also to other criteria such as extension. For example, long major roads with a comparatively small area should be considered as large objects in terms of visual significance. (2) *Objects in sparse region*: Small objects in a sparsely populated region can also be significant. A small town in the desert of central Australia can be more significant than a suburb in a metropolitan area, even when the latter happens to be considerably larger. (3) *Representative objects*: Within a large set of qualifying objects, it might be necessary to select a subset of objects as representatives for the complete set. For example, while it may not be possible to display all the land parcels in an area, some land parcels need to be displayed to indicate the land use. Note that in order to select representative objects, the objects on the boundary of a cluster of objects might be more important than the object inside. The first criterion in our visual significance definition above is a metric constraint, and the last two criteria attempt to model Gestalt constraints in order to give a better overall feeling about the map.

3 Methods

When there are too many objects satisfying the explicit query condition, some techniques need to be developed to filter out the unnecessary objects, while allowing the visually significant ones to pass through. These techniques will make use of implicit conditions derived from display parameters. Broadly speaking, four possible techniques can be used for generalization: (1) *Random selection*: Objects are chosen at random from the source data for display. This solution is discounted because there is no guarantee that the target data set “look” similar to the source data, and there is no consideration about object sizes when removing spatial objects. (2) *Multi-scale database*: Develop a database that store objects with different scale and generalization levels. Instead of developing the methods to transform to any level of scale/generalization, the different levels are pre-coded in the database. This overcomes the speed problem for the generalization process and removes the need to code the transformations to work on any data and with any input. However, it suffers from two major

problems. First, this leads to a huge storage overhead for materializing spatial objects at potentially very large number of scales. This problem can be prohibitive when the spatial data set is very large. Second, this also poses maintenance problems such as maintaining consistency among different representations of the same spatial object when some objects are modified. (3) *On the fly calculation outside of the spatial database*: All objects are fetched into memory and examined on the fly to determine what should be displayed. Although its accuracy and quality would be high, this approach is problematic because of the excessive time required to fetch and examine each object. (4) *On the fly calculation inside of the spatial database*: If visually insignificant objects can be identified inside the spatial database, they will not be fetched and processed outside the spatial database. Inside the spatial database there are much information about object approximation and spatial indexing, which can be used to identify significance of spatial objects without retrieving and processing the full geometry of a large portion of objects. This offers the best potential for speed increase. We focus on this approach hereafter in the paper.

3.1 Object Approximation and Spatial Indexing

Before we discuss different algorithms for data set generalization inside the spatial database, we introduce briefly object approximation and spatial indexing, and spatial data processing strategies based on them.

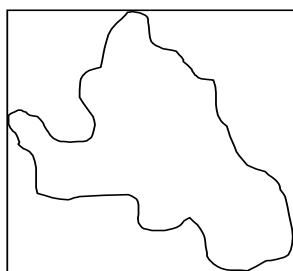


Figure 1: Minimum bounding rectangle

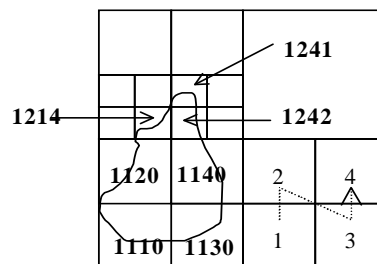


Figure 2: Polygon approximation using z-values

A polygon can be arbitrarily complex. It is a common practice to approximate a spatial object using its *minimum bounding rectangle* (MBR), which is defined as the smallest rectangle that completely encloses the object, see Figure 1. In such a way, an object with any number of points can be approximated using a rectangle that can be represented by only two points (i.e., the lower-left and upper-right corners). The cost of processing spatial data can be reduced significantly by processing their approximations first. For example, two polygons cannot overlap if their MBRs do not overlap; thus, a simple operation on MBRs may eliminate the need to fetch and process full geometry for some polygons, reducing both I/O and CPU cost.

An object often needs to be approximated at a much finer level than its MBR. For example, the MBR of a long road crossing the data space is obviously an inappropriate approximation. The *z-ordering* technique is based on space filling curves. It has been discovered by several researchers independently under different names (see [2] for a survey), and is used in several commercial spatial database systems, such as Oracle [4], as a spatial indexing mechanism. It approximates a given object's shape by recursively decomposing the embedding data space into smaller sub-spaces known as the *Peano cells*. The z-ordering decomposition works as follows. The whole data space is divided into four smaller rectangles of the same size. The four quadrants are numbered as 1 to 4 following certain order (e.g., the z-order in Figure 2). These quadrants can be further divided and numbered recursively. The z-value of a Peano cell can be determined using the following steps: (1) The z-value of the initial space is 1; and (2) The z-values of the four quadrants of a Peano cell whose z-value is $z = z_1 \dots z_n$, $1 \leq z_i \leq 4$, $1 \leq i \leq n$, are $z_1 \dots z_n 1$, $z_1 \dots z_n 2$, $z_1 \dots z_n 3$ and $z_1 \dots z_n 4$ respectively following the z-order. A spatial object can be approximated using a set of Peano cells it overlaps with. The z-values can have different length, as a result of different Peano cell sizes. Clearly, the larger the Peano cells the shorter is the z-value sequence. The maximum number of decomposition level, also called *resolution*, determines the maximum length of z-values. In order to simplify processing, a number of '0's are often appended at the end of shorter z-values to make the length of all z-values identical (i.e., the maximum length). (However, we define the length of a

z-value as the number of non-zero digits it has.) In Figure 2, the polygon has been approximated by 7 polygons, where the resolution is 4. By transforming a two-dimensional object into a set of one-dimensional points (i.e., the z-values), spatial objects can be represented as numbers and therefore can be maintained, for example, by the B⁺-tree [1].

3.2 Algorithms

Three algorithms are chosen to do data set generalization at the database level: (1) *A check for size based on MBR*. The MBR provides a simple way to estimate the size of an object. This estimation can then be used as a measure of the significance of the object. A comparison of an object's MBR area with a threshold provides a way to filter a set of objects based on size, where the threshold can be derived from the spatial extent of all objects to be selected against the size of the display area. This algorithm requires fetching the MBR information about an object, but not the full geometry of the object. This algorithm assigns visual significance based on the size of the object. (2) *A check for size based on z-value information*. Using the z-value that describes an object in a spatial index can be used as an estimate of the relative size of the object. A filter can be developed that takes the input of the length of the z-value and the number of Peano cells it crosses and makes a comparison with these values to determine whether it is visually significant. This algorithm fetches the z-values that describe the location of an object and examine their length as well as the total number of z-values for the object. This algorithm assigns visual significance based on the physical and spatial size of the object. If an object is large physically then the actual footprint of the object is large, an example of this is a national park. If an object is large spatially, then it does not necessarily have a large footprint, but rather extends across a large distance. (3) *A check of isolation based on z-value information*. By dividing the requested region up into fixed size areas, each can be examined to see how many objects exist in that region. If there are below a certain number, all objects in the region are deemed isolated and hence visually significant. For regions with a total number of objects above a certain size, uniform filtering can be performed to reduce the number of objects while maintaining a good impression of the original map. Z-values are used to fetch objects in a given area. From this the amount of objects that exist in this area of space can be determined. This allows a measurement of the isolation of the object to be made. This algorithm assigns visual significance based on the isolation of an object. If a region has below a certain amount of objects all the objects in that region are included. Otherwise uniform filtering is used to sort the remaining objects. This involves including objects in the same position on a grid. Note that this approach differs from the random selection method in two ways: a selection is done only for those crowd regions, and the selection is uniform crossing the region, thanks to the relationship between z-values and the underlying space.

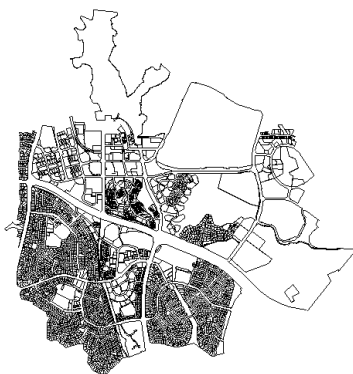


Figure 3: The original map of a Canberra Suburb with 4,500 objects and 50,000 points

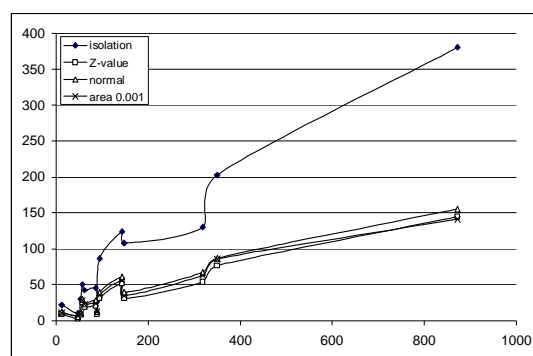


Figure 4: Processing time (seconds) vs. average object sizes for generalization using different methods

4 Performance

In this section we evaluate the three methods discussed above. We consider two aspects: the visual quality of

generalized maps, and the time required producing them. Figure 3 is the original map we used in our test. It contains roads and land parcels in a Canberra suburb, with about 4,500 spatial objects and approximately 50,000 vertices. The original map also contains more than 5,000 textual annotations. Figure 5 shows the generalized map using the three methods. Maps 1-3 are produced by examining MBR areas with different minimum area as the threshold; maps 4-6 by object sizes estimated from z-values using different index levels as the threshold; and maps 7-9 by varying degrees of isolation.

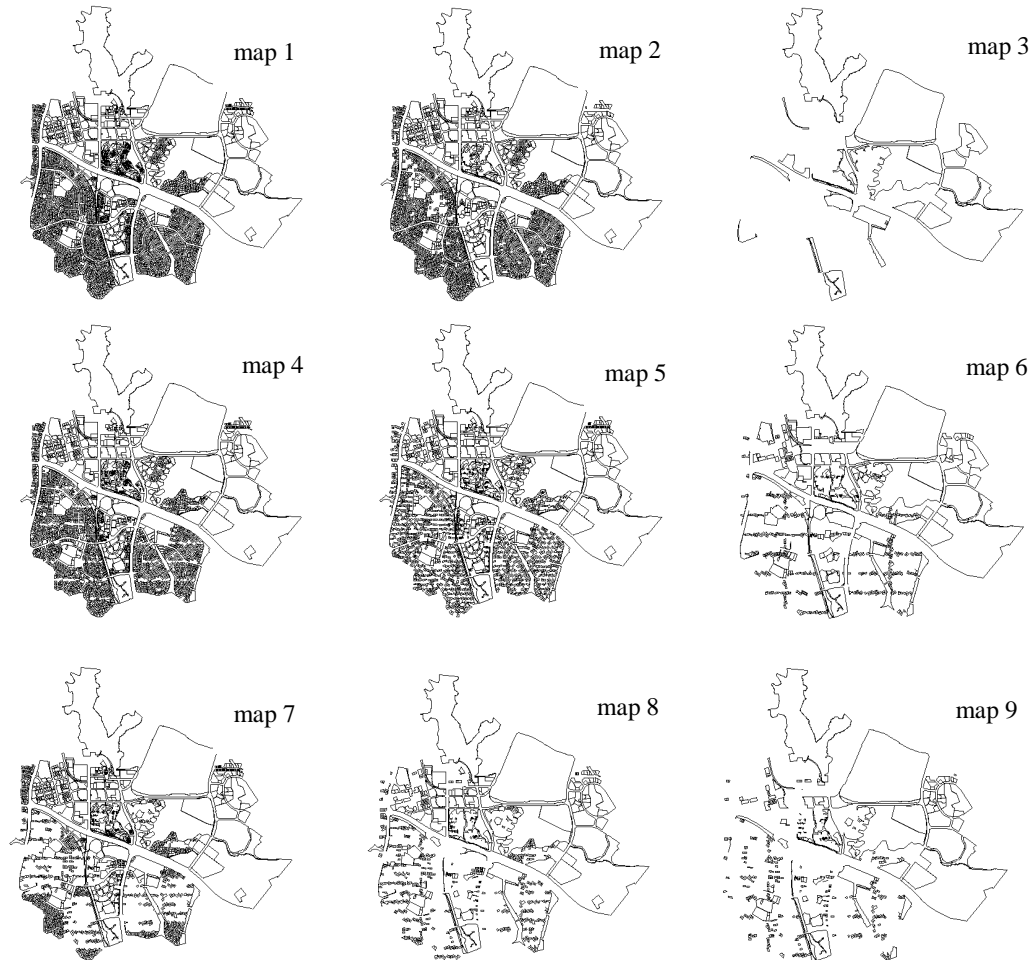


Figure 5: Generalized maps using different methods

The MBR area method does not perform well as far as the quality of the maps it produces is concerned. It either let too many objects through (as shown in maps 1-2), or letting too few object pass (see map 3). It is difficult for this method to work well when there are many spatial objects of similar sizes (such as land parcels in this example). The method that estimates sizes from z-values has achieved the goal of preserving key characteristics of the original map. In particular, map 5 filters out a large number of objects, yet keeps large objects and maintains a good indication of object distribution. Note that objects in a generalized map produced from this method form some type of grid. This is because of the decomposition method used to produce the z-values, as mentioned before, is grid-based. While the isolation method is better than the MBR area method, it fails to maintain the pattern of data distribution in the original map. From a large number of tests we conducted, the z-value based size estimation method outperforms the other two conclusively.

Now we look at the time required producing these maps. Figure 4 shows the response time for the three methods, and the time for retrieving data without generalization (termed as 'normal' in Figure 4). The isolation method always takes considerably more time than other methods. This is because it has to identify density of spatial objects by fetching all objects before processing. The MBR based methods takes nearly as much as time as the method of no generalization. This is understandable because MBR is a simple piece of information every object has. That is, it processes as many objects as the method with no generalization, at little extra cost. The z-value algorithm takes the least amount of time constantly. As mentioned before, z-value is just a simple number, and its manipulation is supported by a B-tree index. In other words, a fast manipulation is possible for this method. Data set generalization is done by such a manipulation, not involving the full geometry of any spatial objects.

In summary, the method of estimating visual significance from z-values is the most efficient algorithm among the three we considered. It requires less time than the method of no generalization due to a smaller amount of target data it produces. It is also able to produce the best quality map, which satisfies all the criteria we used to define visual significance. Moreover, this method takes advantages of a common spatial indexing mechanism, which is supported by several commercial spatial database systems such as Oracle.

5 Conclusions

Currently available web-based spatial applications face major performance problems when it comes to displaying vector data as extracted from a spatial database system. The performance of these systems is well below their potential and there is a pressing need to develop efficient techniques to achieve a performance close to conventional systems. Among many possibilities to improve the performance of web-based spatial applications, the most obvious one is to avoid processing data not used by the applications. In this paper, we have shown that it is possible to provide database support for generalization by identifying irrelevant data already at a very early stage. We have demonstrated that some limited form of generalization can be performed by means of information stored in a typical spatial index. This information allows us to produce a simplified map with satisfactory visual quality with no increase of data retrieval time.

Acknowledgement

This research was conducted in part by the authors when they were in CSIRO. The authors would like to thank David Truffet, Robert Power, Dean Jackson, Mike Clarke, Emma Woolaston, Egon Kuster, Drew Devereux and Richard Leow for their support in implementation and testing.

References

- [1] D. J. Abel. SIRO-DBMS: A Database Toolkit for Geographical Information Systems. *Int. J. Geographical Information Systems*, 4(3):443-464, 1989.
- [2] V. Gaede and O. Günther. Multidimensional Access Methods. *ACM Computing Surveys*, 30(2):170-231, 1998.
- [3] R. B. McMaster and K. S. Shea. Generalization in Cartography. Association of American Geographers, Washington, D. C., 1992.
- [4] Oracle Inc. Oracle 8i Spatial: Experiences with Extensible Database, an Oracle Technical White Paper, May 1999
- [5] The Spatial Archive and Interchange Format, <http://www.elp.gov.bc.ca/gdbc/fmebc/5intro.htm>
- [6] R. Weibel. Generalization of Spatial Data. In *CISM Advanced School on Algorithmic Foundations of Geographical Information Systems*, pages 346-367, 1996.
- [7] R. Weibel. A Topology of Constraints to Line Simplification. In *Proceedings of the 7th International Symposium on Spatial Data Handling (SDH96)*, pages 9A.1-9A.14, 1996.