Road Network Compression Techniques in Spatiotemporal Embedded Systems: A Survey

Amruta Khot
Center for Data Science
University of Washington, Tacoma
1907 South Jefferson, Tacoma WA,
akhot@uw.edu

Abdeltawab Hendawi
Center for Data Science
University of Washington, Tacoma
hendawi@uw.edu
CSE, University of Minnesota
hendawi@cs.umn.edu

Anderson Nascimento
Center for Data Science
University of Washington, Tacoma
1907 South Jefferson, Tacoma WA,
andclay@uw.edu

Raj Katti
Center for Data Science
University of Washington, Tacoma
1907 South Jefferson, Tacoma WA,
rajkatti@uw.edu

Ankur Teredesai
Center for Data Science
University of Washington, Tacoma
1907 South Jefferson, Tacoma WA,
ankurt@uw.edu

Mohamed Ali
Center for Data Science
University of Washington, Tacoma
1907 South Jefferson, Tacoma WA,
mhali@uw.edu

ABSTRACT

The storage and manipulation of road network graphs are critical to navigational and location-based services. The widespread use of GPS devices combined with low-cost storage has enabled portable and embedded systems to handle several spatiotemporal operations against a natively-stored version of the road network graph. However, the increase in amount of map detail data over the years poses several challenges for such systems. In this paper, we highlight the need for adoption of road network compression techniques in embedded geographic information systems. We also provide a technical overview of proposed road network compression techniques.

Categories and Subject Descriptors

H.2.8 **[Database Applications]**: Spatial databases and GIS

General Terms

Algorithms, Performance, Design

Keywords

Compression, Road Network Graph, Digital Maps, Digital Map Generalization, Location-based Services

1. INTRODUCTION

The emerging popularity of GPS-enabled portable and embedded devices has prompted increased usage of navigational and location-based services. Moreover, the quality, and the amount of details a map carries have increased significantly over the past years thereby increasing the payload across devices and servers for GIS applications. At first glance, it may appear that, state of the art storage devices and most portable and embedded systems may afford storing high quality maps. In spite of this, there are many reasons that have triggered researchers to propose solutions for road network compression [6] tailored for such systems.

First, the power and cost constraints on embedded devices limit the storage capacity that embedded devices may have. Meanwhile, the amount of geospatial and non-geospatial information/annotations on a map is growing tremendously. Hence, storing a *compressed* version of the map and its road network graph reduces the storage cost and enables the storage of larger regions as long as the trade-off with power can be balanced. Second, web-based geographical information systems need to give instantaneous responses to user requests for location based services. However, low data bandwidths and big sizes of maps pose a challenge to such responsiveness. Hence, transmitting compressed and/or reduced map sizes can help lower communication costs [4] and enhance responsiveness at the same time. Third, we believe that due to the limited processing capabilities of embedded devices, their computational power needs to be wisely managed. We propose that performing various spatiotemporal operations on top of a reduced size version of the original map would actually lower the computational cost and would lead to near real-time responses. Hence, a geo-streaming flavor of the spatiotemporal operations is achievable.

A road network is represented in the form of a graph structure that has a collection of nodes and edges. The curvature of a road segment is approximated by line segments (or edges) connecting the nodes. The edge weight represents the travel distance or time over the road segment represented by that edge. Road network maps are stored in data files which contain topological and geometrical information. Some compression techniques take into consideration only geometrical information while others consider both topological and geometrical information.

In this paper, we survey various compression techniques for digital maps. We focus on compression techniques that target the road network graphs. One common way of categorizing compression techniques is in two main groups: (1) lossy compression and (2) lossless compression techniques. In lossy compression, certain spatial data is lost permanently as a result of the compression. Lossy compression techniques are acceptable, or

IWGS'14, November 04-07 2014, Dallas/Fort Worth, TX, USA

http://dx.doi.org/10.1145/2676552.2676645

even desired, in cases where not all the details of objects are required to perform the spatiotemporal operation in question [2]. Alternatively, in lossless compression technique, every single data element is recovered when the given map is decompressed. Lossless compression techniques are very important in terms of preserving the topological properties of a map.

The remainder of this paper is organized as follows. Section 2 and Section 3 discuss the lossy and lossless compression techniques for road networks data, respectively. The technical overview concludes with some ideas for use of these compression techniques in section 4.

2. LOSSY COMPRESSION

This section covers the lossy compression techniques for road network map data. Examples of the approaches that fall under the umbrella of lossy compression are: (a) clustering, (b) reference line, (c) map generalization, and (d) a hybrid approach of aggregation and compression. In general, lossy compression techniques discover "similar" chunks of data, create dictionaries on frequently referenced data chunks, and then refer to items in these dictionaries to encode the data. The higher the redundancy in the input data is, the higher the compression ratio is. In this category of approaches some spatial data is lost in the compressed version of the input map and cannot be recovered during the decompression phase.

2.1 The Clustering Approach

The authors in [4] propose a dictionary based compression technique. In dictionary based compression, dictionary entries represent frequent "shapes" of line segments on the map. During data compression, line segments of similar shapes are extracted and represented by a single "representative" line segment. This representative line segment is inserted in the dictionary. Every line segment (among the set of similar line segments) will now point to this representative entry in the dictionary. Upon data decompression, the dictionary is looked up and decompression is done by reverting each line segment back to its representative line segment from the dictionary. Since a set of similarly-shaped line segments are represented by a single entry (i.e., the representative line segment), approximation errors are inevitable between each line segment and the representative line segment.

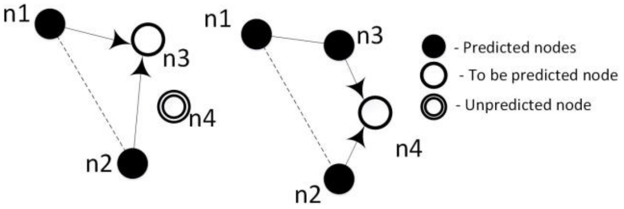
This algorithm separates the topological and geometrical data and the compression is applied only to geometrical data as follows. First, co-ordinates of the uncompressed line segment are considered. The absolute value of the start node is declared as the base point. Then, each subsequent point is represented by a differential vector. A differential vector for point (x_i, y_i) is determined by taking differences (or deltas) between the current point (x_i, y_i) and its previous point on the line segment (x_{i-1}, y_{i-1}) [5].

For example, consider a line with coordinates (1, 1), (2, 3), (3, 2) and (5, 4). The line is encoded using differential vectors as follows: (1,1) , (1,2) , (1,-1) and (2,2) using the formula of Delta(x) = $X_i - X_{(i-1)}$ and Delta(y) = $Y_i - Y_{(i-1)}$. Notice that the differential vectors (after the first point, which is base point) represent the relative shape of the line segment and not the exact location of the line segment. Now, these vectors along with other similar vectors are encoded into the dictionary.

In [4], two approaches for the design of the dictionary are discussed. First, FHM (Fibonacci, Huffman, and Markov) [15] is mainly used for compressing signatures. A static dictionary is built by using Fibonacci series to determine the set of squares for a group of line segments. Each co-ordinate on the line is then encoded by the dictionary entry which lies nearest to the value of the coordinate. Second, a clustering technique is applied to design the dictionary based on the input data. This approach makes use of k-means clustering, for example, to create clusters based on the input data. Then the centroids form the dictionary entries. Each line segment is allocated to a particular cluster and each point references the centroid of the cluster as its base point. The clustering approach for the construction of the dictionary tends to result in lower error boundaries than static dictionary based compression techniques like FHM (Fibonacci, Huffman, and Markov) method.

The delta values that are calculated based on the previous point is helpful for curves having a small number of nodes because the error increases as the number of nodes increase. For longer curves, the following reference line methods seems helpful.

2.2 The Reference Line Approach



(a) Prediction of node n3 (b) Prediction of node n3

Figure 1: Node Prediction during Decompression

A lossy compression algorithm is proposed in [3, 16]. The basic steps of the algorithm can be described as follows.

1. For each polyline in the original map space, a reference line is identified, (usually produced from connecting the two ends of the polyline).
2. The coordinates of that reference line along with its angle from the original coordinate system is used to apply an affine transformation on the points on that polyline.
3. The delta distances in the vertical direction between the intermediate points and the reference line in the new coordinate system are bounded by a predefined error threshold *e*. The selected reference line should keep these deltas within *e*, otherwise, a more representative reference line is selected.
4. In this step, we discuss two modes: (i) In the aggressive mode of the reference line approach [3], which achieves higher compression ratio but less accurate recovery, the original coordinate values for the two ends of the line are stored, along with the number of intermediate points and the error tolerance *e*. At the decompression phase, the algorithm runs two threads of equations to predict the intermediate points of the original curve. The first to restore points closest to the left side of the reference line and the second for the points closest to the right end point of the line. Initially, the two ends of the reference line are leveraged to recover the first point nearest to the left side, e.g., predicting coordinates of n_3 using n_1 and n_2 in Figure 1(a). Then, the two most recent restored points, (from left and right), are used to predict the next point, e.g., restoring n4 using n3 and n2 in Figure 1(b).

(ii) In the less aggressive one [16], (less lossy and less compression ratio), the algorithm stores two vectors of delta distances between each point coordinates $P(x_i,y_i)$, from the origin of the reference line (x_o, y_o) or from its left point, in addition to the two ends of the reference line themselves.

2.3 Map Generalization

Map generalization is a process of reducing the complexity of the map without hampering the topological and structural features [7,12,13]. Generalization operators include simplification and smoothing. One of the most known line generalization and simplification technique is Douglas-Peucker algorithm [10,14].

Douglas-Peucker algorithm is a simple and efficient algorithm. To simplify a polyline, the algorithm starts initially by connecting the first and last vertices of the polyline to form an initial edge. Then, all the vertices between the start and end of the polyline are tested to see whether they are closer to the edge than a certain tolerance factor or not. If all vertices are close enough to the initial edge, the edge in hand is accepted as the approximation of the polyline. If this is not the case, then, the vertex farther away from the initial edge is added to the new approximate polyline and then the algorithm is applied recursively on each edge of the new polyline.

Note that when this approximation is not sufficiently fine, the resulting map can have self-intersecting simplified lines which can affect the topological information of the data. Hence, several other techniques have been proposed to eliminate the self-intersecting problem. Authors in [9] utilize an improved Douglas-Peucker algorithm to avoid self-intersections for any specified tolerance. This improved algorithm keeps the same time complexity of $O(mn)$ where n is the number of input nodes and m the number of output edges as the original algorithm. Saalfeld [8] uses a convex hull to efficiently detect and correct the topological inconsistencies of the polyline with itself and with other polyline characteristics.

2.4 The hybrid aggregation and compression technique

The compression technique in [1,11] has been proposed and integrated with the query processing pipeline of a road network database. Basically, the number of data records is reduced using a line aggregation technique. Then, Huffman compression is applied to achieve a compression ratio on the road network map. In this approach, a map object is a grouping of segments and attributes like the name of the street, e.g., Obj1 = <'Bell st', Seg_1>. All these objects are stored in a database D, e.g., D1 = <Obj1,Obj2,Obj3>. This algorithm uses two operators, one for line aggregation and another for compression.

The line aggregation operator combines multiple map objects into a single map object by concatenating the line segments and by applying an aggregation function over the characters of the street names. An aggregate of D = <Obj1, Obj2, Obj3> is created which is represented by D' = <Obj1', Obj3'>. Here, the Obj1 is combined with Obj2 to create Obj1'.

The compression of the data is done using Huffman coding. In Huffman coding, the frequency f_i of symbol a_i is calculated. These symbols are the coordinates of the line segment. The coordinates of the line segment are mapped to their corresponding binary representation and Huffman coding yields a highly compressed representation of road network map.

When a user asks a query from client machine, results are calculated by the server and returned back to the client. The author discussed two ways in which compression takes place at the server side. In the first technique, line aggregation process is carried out on the "entire" database and the query from the user is passed to the aggregated. Then, the query result is compressed using the compression operator and send back to the client. In the second technique, the query is fed to the original database and then the query result is aggregated, compressed (using the line aggregation and compression operators, respectively) and returned to the user. In the first technique the whole database is aggregated while in the second approach aggregation is only performed on the query result.

3. LOSSLESS COMPRESSION

In lossless compression technique, every single data element is recovered when the given map is decompressed. Lossless compression techniques are very important in terms of preserving the topological properties of a map after decompression. In this paper, we discuss the most common two categories of lossless compression, namely, predictive approaches and topological and geometrical approach.

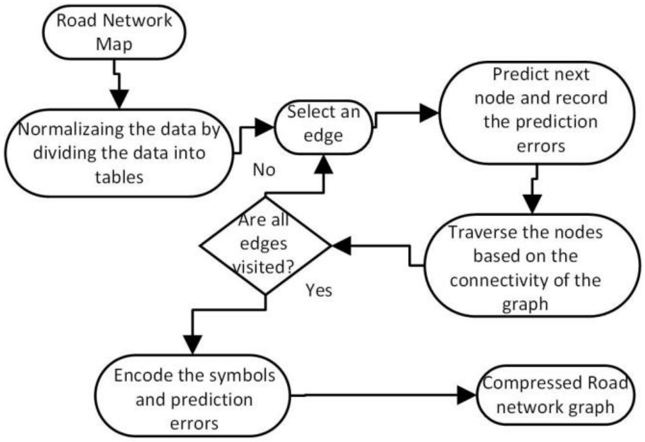

Road Network Map
Normalizaing the data by dividing the data into tables
Select an edge
Predict next node and record the prediction errors
Traverse the nodes based on the connectivity of the graph
Are all edges visited?
No
Yes
Encode the symbols and prediction errors
Compressed Road network graph

Figure 2: Steps of Predictive Compression Approach

3.1 Predictive approach for road network compression

The proposed approach in [2] accomplishes a lossless compression technique taking into consideration the topology of the map. The basic idea is to navigate through the given road network map edge by edge and predict the next node by using a prediction model. The errors produced by this prediction model will then be compressed using entropy coding methods. This compression scheme encodes a node using less number of bits than originally required.

The steps to compress a map are shown in Figure 2. This algorithm starts with organizing the road network map data into three tables. One table records the edge Id, start node of the edge and end node of the edge. The second table contains the edge Id and the intermediate nodes belonging to that edge. The third table maintains the information of the co-ordinates of each node. This algorithm does not separate the topological data from the geometrical data during compression.

The algorithm navigates through the spanning tree of the road network graph. A start edges is selected if it satisfies the minimum number of nodes required by the predictor to predict the next vertex. The edges having less number of vertices than required for the prediction are not compressed.

The algorithm defines two prediction models: (a) a node based linear prediction model and (b) an angle-length based prediction model. The node based linear prediction model predicts the next node based on the previous two nodes, hence the minimum requirement of each edge is two nodes. The angle-length based prediction model requires a minimum of four nodes. It predicts two turn angles and the length of the next turn for the next node.

3.2 Topological and Geometrical information compression

The authors in [6] propose an approach which considers the compression of road network nodes as well as the shapes. The basic idea is to divide the space into a number of small cells using an efficient partition spatial index such as Quadtree, KD-tree, or R-tee. Then, for the portion of the road network graph in a cell c, the algorithm finds a reference point from which the location differences to other nodes will be computed and stored in less number of bits. For example, the road network graph in Figure 3 is partitioned into three parts $\{P_1, P_2, P_3\}$. In P1, the node n1 can be the reference point for the remaining nodes n_2, n_3, n_4. For all shapes in the cell c, they are clustered into different group of similar patterns. Then a reference shape is computed, or picked up from existing ones, to represent the common theme in each cluster. After that, each shape is stored as delta values to the reference shape. This road network compression approach achieves more than 30 percent of reduction compared to other compression techniques available for the European road networks.

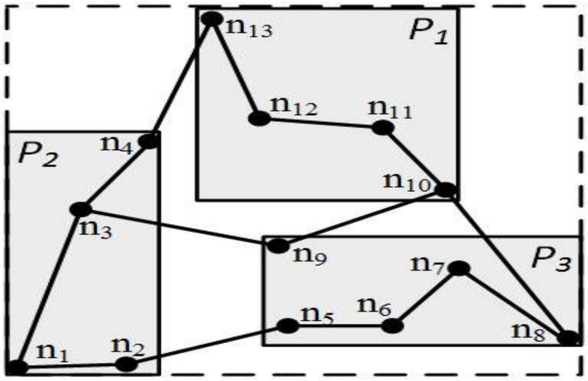


Figure 3: Partitioning of Road Network Space

4. CONCLUSION

In this paper, we highlighted the importance of storing a “compressed” version of the road network map inside spatiotemporal embedded systems. More specifically, compression achieves higher storage utilization, reduced communication cost and lower response times. Hence, portable and embedded devices can natively implement a *geostreaming* flavor of spatiotemporal operations that achieve near real time output. We overviewed various lossy and lossless compression techniques for road networks and highlighted future challenges.

5. REFERENCES

[1] Ali Khoshgozaran, Ali Khodaei, Mehdi Sharifzadeh, and Cyrus Shahabi. A hybrid aggregation and compression technique for road network databases. Knowledge and Information Systems, 17(3):265–286, 2008.

[2] Zongyu Zhang. Vector road network compression: a prediction approach. In Proceedings of the American Society for Photogrammetry and Remote Sensing Conference, ASPRS, Reno, Nevada, USA, May 2006.

[3] Alexander Akimov, Alexander Kolesnikov, and Pasi Franti. Reference line approach for vector data compression. In Proceeding of the IEEE International Conference on Image Processing, ICIP, pages 1891–1894, October 2004.

[4] Shashi Shekhar, Yan Huang, Judy Djugash, and Changqing Zhou. Vector map compression: a clustering approach. In Proceedings of the ACM International Conference on Advances in Geographic Information Systems, ACM GIS, pages 74–80, VA, USA, November 2002.

[5] Williams, H.E., Zobel, J.: Compressing integers for fast file access. The Computer Journal 42(3), 193–201 (1999).

[6] Suh Jonghyun, Jung Sungwon, Pfeifle Martin, Vo Khoa T, Oswald Marcus, and Reinelt Gerhard. Compression of digital road networks. In Proceedings of the International Symposium on Advances in Spatial and Temporal Databases, SSTD, pages 423–440, Massachusetts, USA, July 2007.

[7] Nabil H. Mustafa, Shankar Krishnan, Gokul Varadhan, and Suresh Venkatasubramanian. Dynamic simplification and visualization of large maps. International Journal of Geographical Information Science, 20(3):273–302, 2006.

[8] Alan Saalfeld. Topologically Consistent Line Simplification with the Douglas-Peucker Algorithm. Cartography and Geographic Information Science, 26(1):7–18, 1999.

[9] Shin ting Wu and Mercedes Roco Gonzales Mrquez. A Non-Self-Intersection Douglas-Peucker Algorithm. In Brazilian Symposium on Computer Graphics and Image Processing, SIBGRAPI, pages 60–66, Ouro Preto, Brazil, August 2003.

[10] David H. Douglas and Thomas K. Peuker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. The International Journal for Geographic Information and Geovisualization, Cartographica, 10(2):112–122, 1973.

[11] Ali Khoshgozaran, Ali Khodaei, Mehdi Sharifzadeh, and Cyrus Shahabi. A multi-resolution compression scheme for efficient window queries over road network databases. In the International Workshop on Spatial and Spatio-temporal Data Mining, pages 355–360, December 2006.

[12] Michela Bertolotto and Max J. Egenhofer. Progressive transmission of vector map data over the world wide web. Geoinformatica 5(4):345–373, 2001.

[13] Barbara Buttenfield. Transmitting vector geospatial data across the internet. In Proceeding of the International Conference on Geographic Information Science, GIScience, pages 51–64, 2002.

[14] Xavier Barillot, Jean-François Hangouet, and Hakima Kadri-dahmani, Generalization of the Douglas and Peucker Algorithm for Cartographic Applications, in proceedings of the 20th International Cartographic Conference, ICC, Beijing, China, August 2001.

[15] D. Solomon, Data Compression: The Complete Reference, 2nd edition, Springer-Verlag, 2000.

[16] Minjie Chen, Mantao Xu, and Pasi Frnti. Fast dynamic quantization algorithm for vector map compression. In Proceeding of the IEEE International Conference on Image Processing, ICIP, 2010.