Effective Topological and Structural Compression of GML Coordinate data

N.N.Harshita[1] & K.S.Rajan[2]
Lab for Spatial Informatics,
International Institute of Information Technology, Hyderabad.
[1] harshita@research.iiit.ac.in [2] rajan@iiit.ac.in

Abstract

Spatial data infrastructure (SDI) is a framework which mainly focuses on bringing together geospatial data providers and agencies to share data and services thus reducing the redundancy in data collection and improving the planning and decision making process. Geography mark-up language (GML) has largely come to stay as the standard for these SDIs. GML is mainly used for storage, exchange and querying of geographic data. The standard format for GML data is text leading to large data storage sizes, which if reduced/compressed will improve both the data storage and its management. Also, GML data compression will aid in higher information transfer and allow the users for faster access & quicker visualization. Recent works on GML compression have largely focused on the structure of data, while this paper proposes a method involving both topological and structural characteristics of the data. The topological relationship in the data is exploited for reduction in its size by eliminating the data reoccurrences. In addition, to achieve a lossless compression and for easy storage and retrieval of the complete data, a tree based data-structure is proposed to represent the coordinates which further helps in reduction of the overall data size. The experimental results show that this method achieves a compression percentage of 50% to 70% for datasets ranging from 244 to 16,000 KBytes.

Keywords: Compression, GML, Topology, Structure

1 INTRODUCTION

In recent times with the increasing geo-awareness, e-governance and decision making in a range of disciplines from environment to economics; individual users to corporations, the emphasis on developing Spatial Data Infrastructures (SDI) is evident at all levels - from local to national to global. SDI comprises of a framework for decentralised way of data storage and management for both data providers and the users. SDI is envisaged to provide services to seamlessly integrate the various thematic layers or information and make the appropriate content available as requested by the users in near-real time. To achieve this, interoperability of the data and efficient transfer mechanisms are needed. Data interoperability defines the ability of data to be integrated seamlessly in different systems and is an integral part of SDI. eXtensive Mark-up Language(XML) is a technology for implementing the schematic interoperability. It facilitates data storage and transfer in a manner that enables seamless sharing of data regardless of application or hardware platform (Otieno et. al., 2003). GML or Geography Mark-up Language is an XML based encoding standard for geographic information developed by the OpenGIS Consortium (OGC) and is only concerned with the representation of geographic data content (Peng et. al., 2004). GML serves as a modelling language for geographic systems as well as an open interchange format for geographic transactions on the Internet. (http://www.opengeospatial.org/standards/gml). GML is used in the request and response messages of WFS (Web Feature Service) and other Geo-Web services, and is part of the standard service for accessing geographic feature data (Lake et. al., 2004). GML represents both simple and complex geographic features using a data description language with strict hierarchical data structures and it facilitates data search and discovery on the

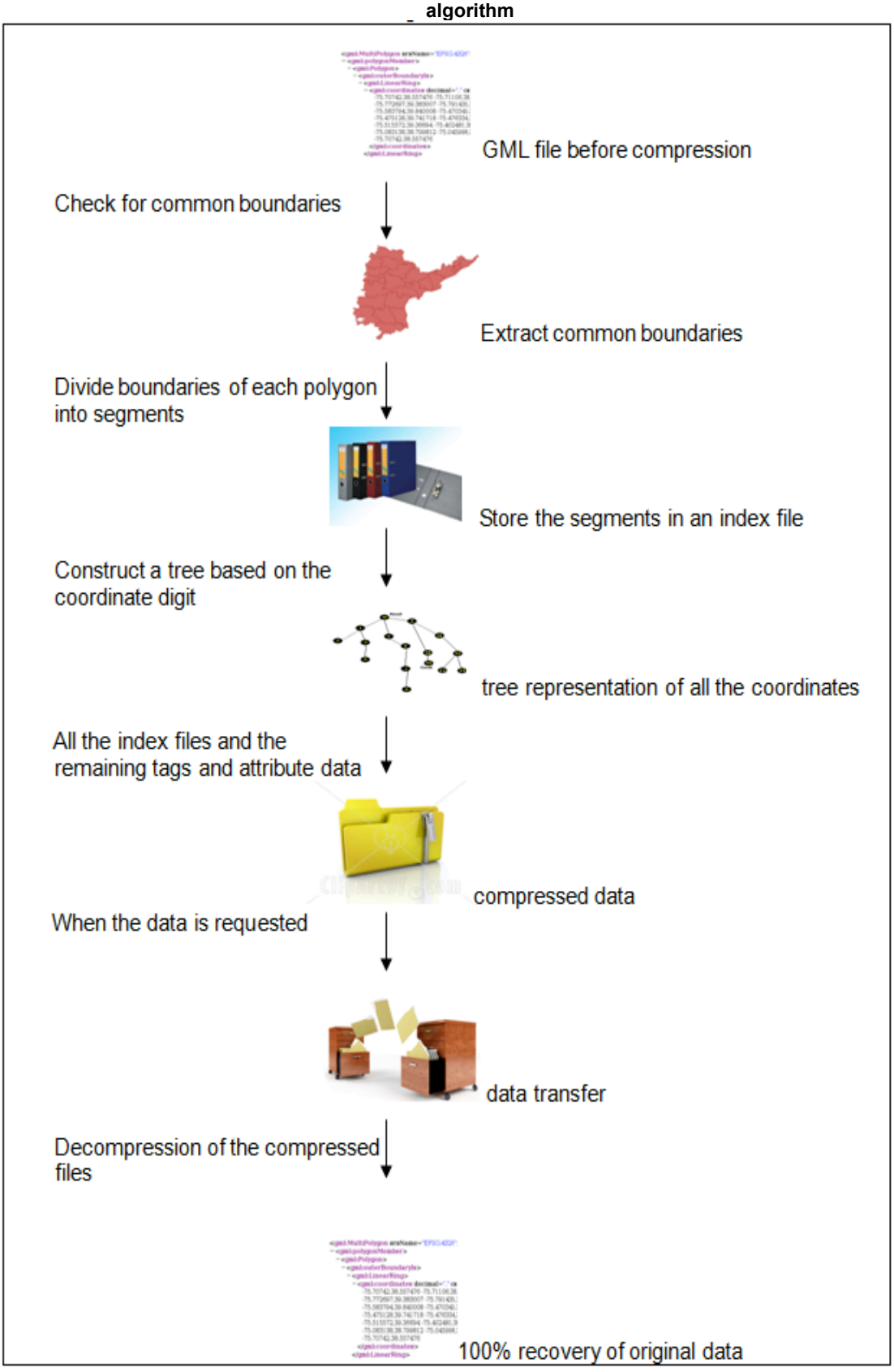
Web. With the increase in Web-GIS systems and the promotion of SDI in recent times, most of the geographic data is stored, transferred and exchanged in GML, which is further expected to increase many fold. (Brisaboa et. al., 2006). An important challenge to this is - how to store and manage the large amounts of GML data? In addition, the data sizes will affect the Geospatial data utility and usability due to the varied data transfer rates that a multitude of systems offer the users. If the size of this data can be reduced, both transfer and viewing of the data will become faster, easier and also affordable in some parts of the world.

With increase in the amount of data, the number of users of the data has also increased considerably. This will lead to more data access requests and I/O transfers, which has the potential to overload the geo-servers. Hence, it is important to deliver the data efficiently and effectively across the network and reduce the sever I/O access to a considerable extent. One of the possible solutions is the use of distributed databases and distributed networks including mirror sites. The other solution is reduction in the size of the data, i.e., GML data compression. There are two types of GML data compression – a domain specific compression dealing with special kinds of data like cadastral data, etc., and the other is domain independent compression or generic compression. To a certain extent both these methods have been explored, but the inherent topological relationships that geospatial data has have been largely ignored and the syntactic nature of the data has been focused. For instance in the GPress algorithm (Guan et. al., 2008), the document structure and the coordinate data are separately treated for compression, using a float point data compressor for the latter, and the *gzip* algorithm for the former. Similarly, the work by Brisaboa et. al., (2006) has also mostly used textual non semantic compression like removing spaces and replacing the keywords with codes. For the coordinate data, the large number of significant number repetition is avoided by representing each coordinate as a difference to the earlier one and then the various text compression techniques have been applied and compared. The GHIC algorithm (Zhang et. al., 2008) has a domain specific approach and uses clustering techniques and compresses data based on its isomorphic characteristics, and exploits the tags for encoding. While, GQComp (Dai et. al., 2009) focuses on the document structure and builds a feature based tree for the spatial data but has no focus on the topology and structure of the coordinate data. Google Earth API uses its polyline encoding algorithm to compress the coordinate data and reduces the data size by rounding off the fractional part based on the zoom level leading to a lossy compression (http://code.google.com/apis/maps/documentation/polylinealgorithm.html). Some projects use normal file compression techniques for compressing the GML files, for example the Karnataka state spatial data infrastructure project (http://www.karnatakageoportal.in/KSSDI/home.html) has developed a web-based geo portal that uses GNU gzip compression tool to compress and store the GML data files and does not exploit any of the inherent properties of GML data structure or content.

As can be seen from the current works, all of these do not exploit the topological characteristics of the geo-data and are more focused on the syntactic nature of the data, more so on the data presented in the GML files. Also, the above compression and decompression methods are server-centric and not at the client or browser end, thus having an impact only on the data storage and not on the data transmission or bandwidth. The main focus of this paper is to address the compression of GML data such that it improves both the data storage and strengthens the client-server architecture of the SDI, mainly data transfer and bandwidth issues. The later is addressed by building a client level browser-plug-in that can seamlessly uncompress and display the data to the user. It is important to note here that the coordinate data is a major part of the geospatial data irrespective of the domain, as it defines the location and boundaries of each and every area and feature of the data (see Table.1) and hence has to be the central part of any GML data compression. Also, a hierarchical tree structure is introduced to manage the coordinate data with emphasis on the

topographical characteristics of the data, which has a structure similar to the GML data file. A lossless compression approach is adopted to recover the original data after decompression.

Figure 1: Overall Flow Chart of the GML Compression/de-compression algorithm



In addition, the compression is done at the server level and the decompression is done at the client level for seamlessly less effort in receiving and viewing the data using simple or lean clients and reduce load on the server.

Overview of the paper: The next section describes the approach in detail for compressing the data. Experimental results section describes the input data on which this method is applied and also the results showing the amount or percentage of compression achieved with respect to various aspects of the data. The Conclusion section discusses the benefits and drawbacks of this method and also the future work.

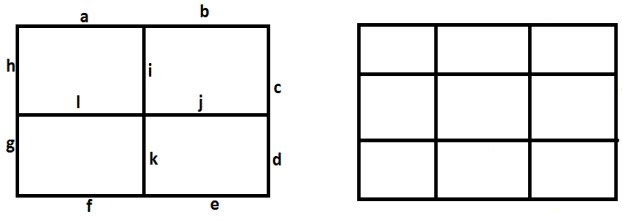
2. PROPOSED METHODOLOGY

The proposed methodology comprises of mainly two steps – (i) the common boundaries extraction and edge indexing; and (ii)coordinate data management using a tree structure. The first step mainly focuses on the occurrence of common boundaries based on the feature's topological relationships, while the second step introduces a tree structure for eliminating the repetition of digits in the coordinates. The work presented here focuses on the polygon data, since they exhibit the adjacency property which is exploited here. Figure 1 shows the overall flow of the proposed methodology.

2.1 Common boundaries extraction

The topology of the data is exploited by identifying and extracting the common boundaries that repetitively occur in more than one polygon. Take for instance, Fig.2 (a) which represents an area of interest and has 4 polygon features. If coordinates of the edges of each polygon are stored separately, then 4 out of the 16 edges (viz., i, j, k and l) which are common to two polygons are stored more than once leading to repetition of these in the storage media. Instead, if each polygon is divided into arcs or edges, then just 12 edges and its corresponding arc index need be stored. Similarly, for the same area of interest with 9 polygon features as shown in Fig.2 (b), only 24 edges are enough to describe the 9 polygons with 36 edges. This leads to a reduction in the number of coordinates to be stored. As can be seen from this example there is a reduction in data size from one-fourth (25%) to one-third (33%) with increase in number of connected polygons.

Fig.2 An area of interest consisting of – (a) 4 polygons; and (b) 9 polygons



The following are the steps followed for common boundaries extraction and generation of index file for a given GML file, with adjacent polygons, is as follows –

Step 1:- Initially, all the common boundaries or edges of the adjacent polygons are identified by searching for repeated common sequences in the GML file.

Step 2:- The entire boundary of each polygon is divided into *arcs*. For each polygon, if a set of edges are common to two polygons they together form an arc and the remaining continuous edges of the boundary which are not common form the other arcs. So each polygon is represented by a set of arcs.

Step 3:- All the common arcs are indexed and stored in an index file, and these index value are inserted into the GML data file. In the index file, all the arcs starting with the same coordinate are bunched and stored together.

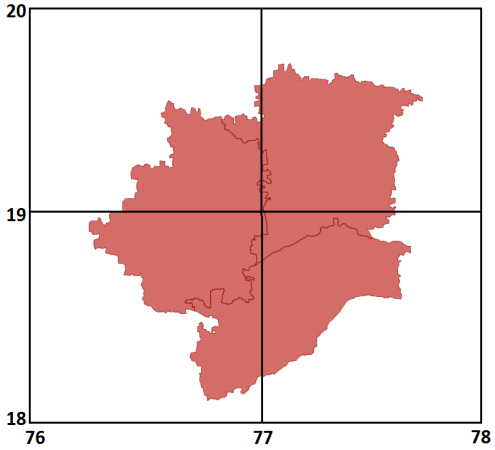
2.2 Coordinate Data Management – a tree-based data structure

After the extraction of common boundaries, the rest of the file contains non-common edges, tags and other attribute data. In space, the edges of these polygons are locally occurring leading to the observation that many characters/digits in the coordinate values are repeated. Squeezing out these repeats can lead to a good reduction in size as only a small part of the coordinate values differ significantly. But for a lossless recovery of the original data, these values have to be recovered at the user end. To achieve this, a tree-based data structure for storing the coordinate values is proposed. As the GML data file also follows a tree-based structure, this compression approach yields itself to be included in it, which makes traversing through the data and coordinates easier, leading to a quicker decompression of the compressed data. Since more the similarity between the coordinates more optimal the tree becomes, it is imperative to choose the region blocks appropriately according to the scale of the input data represented by their respective tree-data storages. The block size depends on the scale and resolution of the GML data that is to be compressed and is generated suitably. If the resolution of the data is high the block sizes are smaller, whereas if it has low resolution the block size is larger. Every node of the tree has a value and a child list associated with it. In addition, all the leaf nodes store an extra vertex index for finding the corresponding coordinate in the other dimension. The algorithm is given below and corresponds to Fig 3 and 4 below.

Step 1:- To generate the tree-structure, identify the root node and label it with the respective block id.

Step 2:- The first level child node of the root node stores the integral part of the coordinate, which is uniquely different for the respective dimensions. As can be seen from Fig 3, the region block has two branches – 76 and 77, along the x-axis.

Fig.3. A representative sample region from the district map of India.



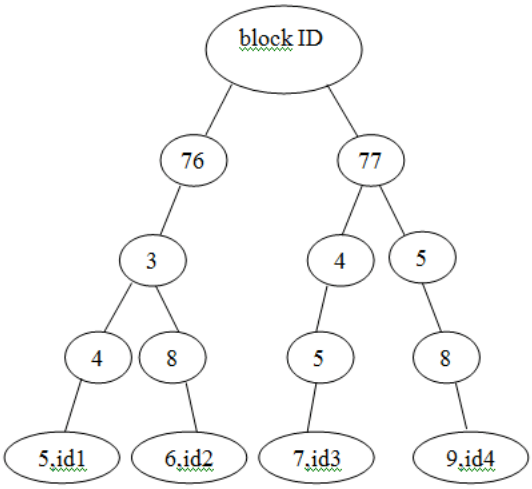
Step3:- The digits of the fractional part i.e., digits after the decimal point are now inserted into the next subsequent child nodes (or levels) in a way such that the combination of the upper nodes in its hierarchy are identical while it is different at that level and so on till the last leaf node. The leaf node also contains a vertex index for easy retrieval and reconstruction of the data. All the data is stored uniquely such that no node is stored more

than once.

Let's take for instance, a part of Fig 3 represented by a region containing polygons, which has a set of X-coordinates (76.345, 76.386, 77.457, 77.589 and so on...) belonging to it. The root of the tree of this polygon will have the area block ID stored and 76 goes into the first level. Then 3 becomes the child of 76, and similarly 4 is the child of 3 and 5 the child of 4. The vertex index is associated with the leaf node. Now, consider the next coordinate 76.386. Its integral part is 76 which is already present in the first level, so it is not inserted. Similarly the first digit of the fractional part 3 is already in the child list of 76 and no action is taken. The next digit 8 is inserted in the child list of 3, while the last digit 6 becomes the leaf node and a child of 8. Fig 4 below shows a part of the data tree that is stored for this region of interest. Similarly, Y-coordinate tree is built with the same root node value. The vertex index value stored in the respective leaf nodes of the X and Y-coordinate trees are used for retrieving the data while decompressing these compressed files.

This tree structure also can easily be updated for any changes in the number of vertices - addition and deletion of polygon vertices and edges, by traversing the tree and making suitable modifications at the right level/child of the tree. Suitable index updates can be performed to reflect these changes, thus providing for flexibility in its adoption for even transaction data handling, though the later is not in the scope of this paper.

Fig.4. A sample representation of the tree-based data structure for coordinate storage



A similar method is followed for the index files and trees are constructed for each key in the dictionary based on the same data similarity perspective. The index file is divided into two, one file with X-coordinates and other with Y-coordinates. The tree is an n-tree which can have n-children at each level, where n lies between 0 and 9. But the number of nodes in the first level children may vary according to the size and extent of the area that is being dealt with. The coordinates of the common boundaries in the index file are also stored in a tree structure and they can be directly embedded in the tree to which they belong for retrieving the original data during decompression.

3 RESULTS AND DISCUSSION

To evaluate the performance of the proposed method, it was applied on a range of GML data file sizes. These GML data files were created from the district boundary dataset of India covering various geographies. Table 1 lists all the data file along with their respective characteristics - total number of polygons, vertices, size of the file, size of the coordinate

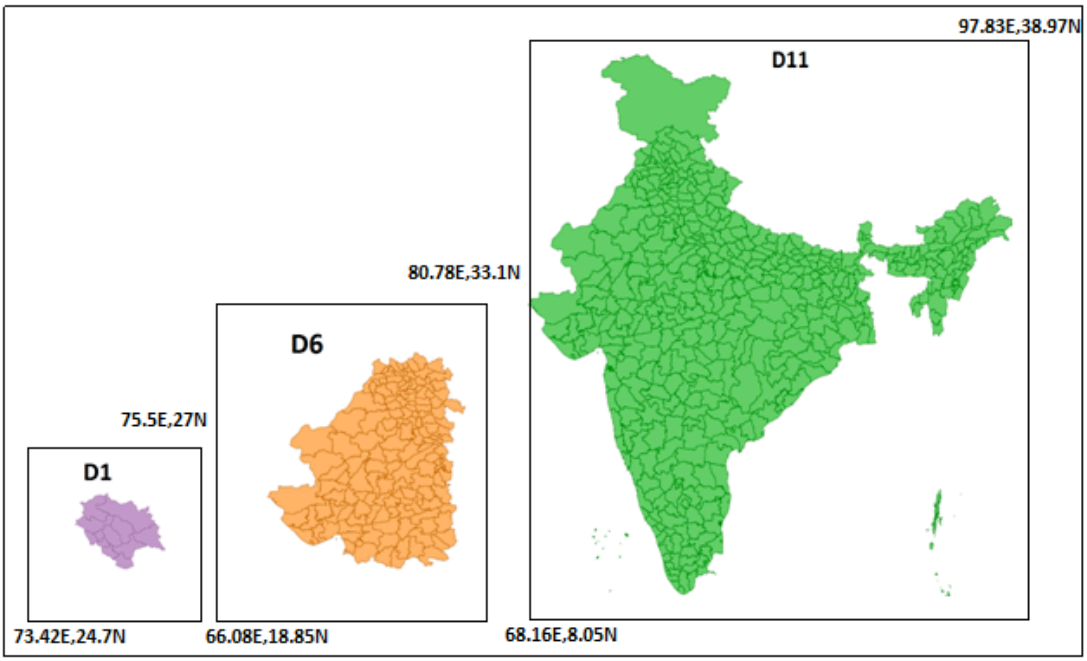
data and the percentage of the coordinate data present in the file.

Table.1. List of the experimental data set with its characteristics

GML File name	No. of polygons	No .of vertices	GML file size (in kilobytes)	Coordinate data size(in kilobytes)	% of the coordinate data in file
D1	17	5460	244	236	96.72
D2	13	9457	416	392	94.23
D3	24	17163	768	740	96.35
D4	29	26578	1200	1124	93.67
D5	72	52217	2300	2152	93.57
D6	117	84999	3600	3464	96.22
D7	145	113833	4900	4689	95.69
D8	185	152421	6500	6150	94.61
D9	267	204000	8600	8205	95.40
D10	351	263678	12000	11568	96.4
D11	468	357103	16000	15450	96.56

As can be seen from fig.5, the data files chosen show a range of spatial extent and topological relationships, with the polygons being largely irregular with multiple edges and each edge having large number of vertices

Fig.5. Sample view of some of the data files



Both the proposed algorithms - common boundary extraction and the coordinate data tree generation algorithm were applied to these GML data files, separately and jointly to test the levels of compression achieved. The results are tabulated in Table2. Independently, each of these methods can lead to around 30 to 40% reduction in the data sizes and together upto 70%. It can be noted here that when both the methods are applied, the reduction percentage is less than the sum of their individual compressions. This is due to the additional overhead of storing indexes of the common boundaries and other relevant data for making completely lossless decompression possible.

Table2. shows the compression percentage obtained for the proposed approach.

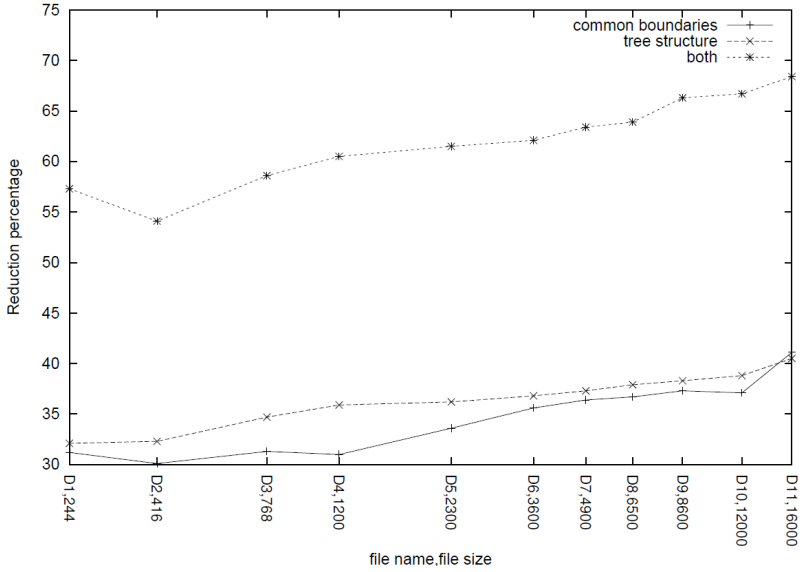
GML file name	% of compression using common boundaries approach	% of compression using coordinate data tree approach	% of compression when both are applied
D1	31.2	32.1	57.3
D2	30.1	32.3	54.1
D3	31.3	34.7	58.6
D4	31	35.9	60.5
D5	33.6	36.2	61.5
D6	35.6	36.8	62.1
D7	36.4	37.3	63.4
D8	36.7	37.9	63.9
D9	37.3	38.3	66.3
D10	37.1	38.8	66.7
D11	41.1	40.5	68.4

3.1 Effect of data file characteristics on the compression

File Size and Compression ratio

Though intuitively, increase in data file size should lead to increased compression, it can be seen from Fig 6., that it is not so. Data files D1 to D4 show almost similar compression ratios for the common boundaries approach, though the tree based approach responds to the file size. This indicates that the non-geographic content (like tags, etc) dominates in small files and needs to be considered for effective compression. But, it can be seen that the combined application of the two takes care of some of these ill-effects of the common boundary approach, and one of the reasons for this may be due to the role that the number of polygons have on the compression. Though at D4 and D10, we find a dip in compression, the overriding effect of increasing number of polygons – 20% to 30% (compared to the preceding data file) as can be seen from Table 1, is having a positive effect in improving compression of the tree-based approach, which is not the case for D2 where there is a drop in the number of polygons.

Fig .6 : Percentage of Compression in relation to the file size of the data

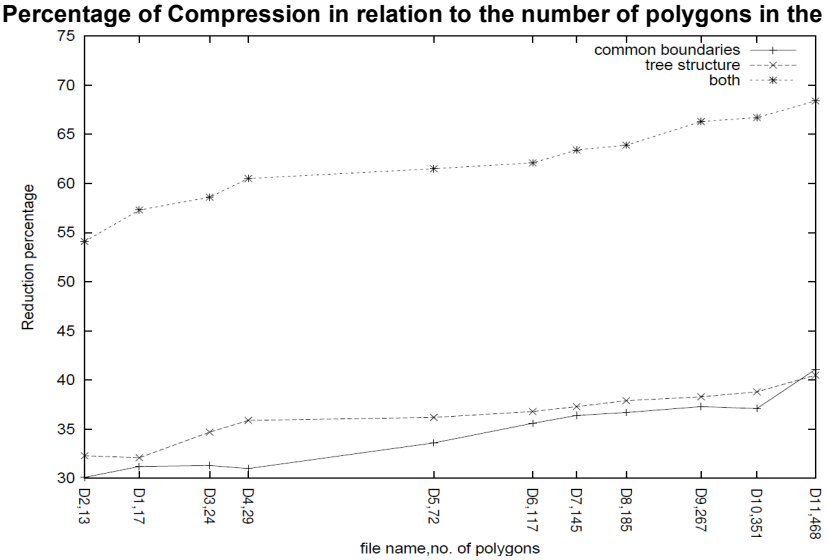


Number of Polygons and Compression ratio

It is clear from the Fig 7, that with the increase in the number of polygons, the compression ratio increases, indicating that the topological relationships become a crucial factor in

identifying and eliminating the data repeats. Also, the number of polygons per unit area helps the data tree approach though the common boundaries may not be affected, as seen from data files D4 and D10. But, with increase in polygons, the data tree structure will also increase leading to more overheads than to the common boundaries index. This may be one of the reasons for not having sharp increases in compression in the former approach.

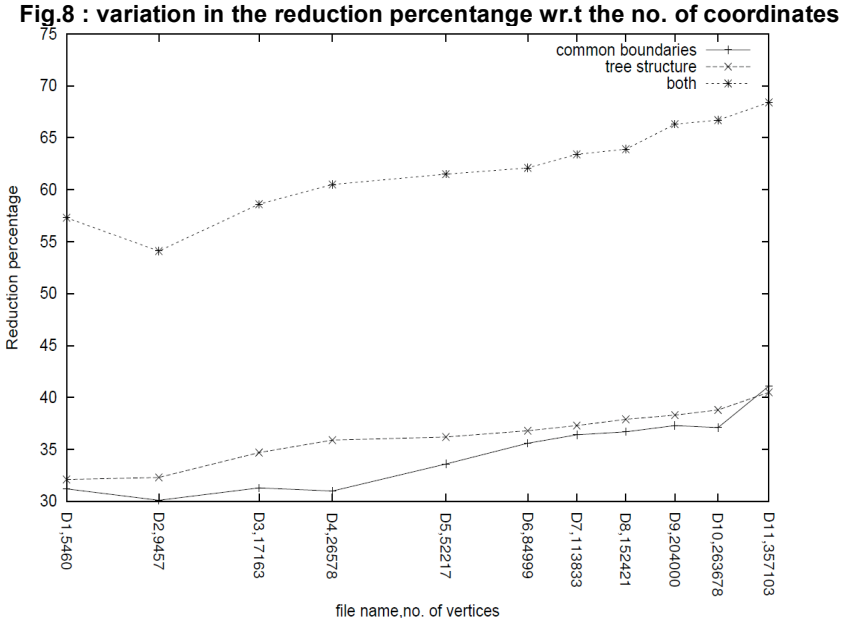
Fig.7 : Percentage of Compression in relation to the number of polygons in the data files



Number of Vertices and Compression ratio

As can be seen from Table 1, the dataset files have an increasing number of vertices and Fig 8 shows a plot of the compression ratio with respect to the number of vertices. Though there is a good increase in the compression ratio for the common boundaries approach, the data tree approach increases more slowly. This can be because of the geographic spread of the data, as it leads to the creation of multiple region blocks in the data tree. The later will tend to increase the overhead data leading to a marginal increase in the overall compression. It is to be noted here that since most of the users of spatial data will access relatively local information, this may not become a significant factor other than the amount of geographic content itself in these datasets.

Fig.8 : variation in the reduction percentange wr.t the no. of coordinates



4. CONCLUSION

SDI is a repository for both spatial and non spatial data sets, which is increasing in content and size rapidly. There is a need to reduce the size in order to be able to manage and analyse this data. The SDI framework largely uses GML as its format for data storage and management. This paper has demonstrated a methodology in exploiting the topology and structure of the coordinate data on the compression potential of a standard GML data file. Since coordinate data occupies more than 90% of the total file size (see Table 1), it is clear that more the number of coordinates more is the compression and which has been exploited in most of the approaches included in this paper. The data tree approach helps in reconstructing the data without any loss and this is a great advantage of this approach. In addition, the adjacency characteristic has helped in reducing the data further. It is needed to explore and exploit the other locally explicit topological relationships for building more user friendly and efficient compression. Thus, data storage and handling becomes easier.

An added advantage of this compression method is that the decompression can be done at the client end using a plug-in, thus reducing the load on the server. In addition, along with data compression, there is a need to focus on proper handling of data updates and reducing the complexities involved in this method by making modifications to the compressed format itself rather than uncompressing the data. The suggested approach also yields itself to building an effective data query system over the compressed data leading to faster responses to the queries as the search space (data size) is much smaller. Further work will focus on building a generic framework for GML data compression and query system. Since the current approach has not focussed on the encryption of these data, as data security is expected to become an important concern of SDI, it may also be needed to be incorporated in future.

5 REFERENCES

Anonymous, OpenGIS Geography Markup Language (GML) Encoding Standard, http://www.opengeospatial.org/standards/gml, [accessed 1 April 2010].

Anonymous, Karnataka Geo portal, http://www.karnatakageoportal.in/KSSDI/home.html, [accessed 1 July 2010].

Anonymous, Encoded Polyline Algorithm Format, http://code.google.com/apis/maps/documentation/polylinealgorithm.html, [accessed 1 April 2010]

Brisaboa, N. R., Farina, A., Luaces, M. R., Viqueira, J. R., Parama, J. R(2006). On the Compression of Geography Markup Language, Advances in Computer Science and Engineering, 19, pp. 93-104.Mexico, 2006.

Dai Qiang, Shuliang Zhang, Zhihui Wang (2008). GQComp: A Query-Supported Compression Technique for GML, *Ninth IEEE International Conference on Computer and Information Technology,* Vol. 1, pp.311-317.

Guan Jihong, Shuigeng Zhou, Yan Chen (2008). An Effective GML Documents Compressor, *IEICE – Transactions on Information and Systems,* Vol. E91-D, Issue 7, pp.1982-1990.

Lake Ron, David Burggraf and Milan Trininic (2004). Geography Mark-Up Language (Gml): Foundation for the Geo-Web Approach, John Wiley & Sons Ltd.

Li Yuzhen , Jun Li and Shuigeng Zhou (2005). GML Storage: A Spatial Database Approach, Springer Berlin / Heidelberg.

Otieno Fredrick Tom (2003). “*Developing XML biodiversity geo-database in the context of corporate SDI at Department of Resource Surveys and Remote Sensing (DRSRS), Kenya”*, thesis submitted to the international institute for geo-information and earth observation.

Peng Zhong-Ren and Chuanrong Zhang(2004), The roles of geography markup language (GML), scalable vector graphics (SVG), and Web feature service (WFS) specifications in the development of Internet geographic information systems (GIS) , Journal of Geographical Systems, Vol. 6, pp. 95-136. Springer Berlin / Heidelberg.

Zhang Haitao, Shuliang Zhang, Guonian Lv (2008). "GHIC: A GML Compressor with Holistic Isomorphic Characteristic”, *2008 International Workshop on Education Technology and Training & 2008 International Workshop on Geoscience and Remote Sensing, December 21-22 2008,* vol. 1, pp.281-284.