The Evolution of Geography Markup Language (GML) Compression Model

Muhammad Mahmud, Azman Yasin, and Mazni Omar

Universiti Utara Malaysia (UUM), Malaysia, s95772@student.uum.edu.my, {yazman, mazni}@uum.edu.my

ABSTRACT

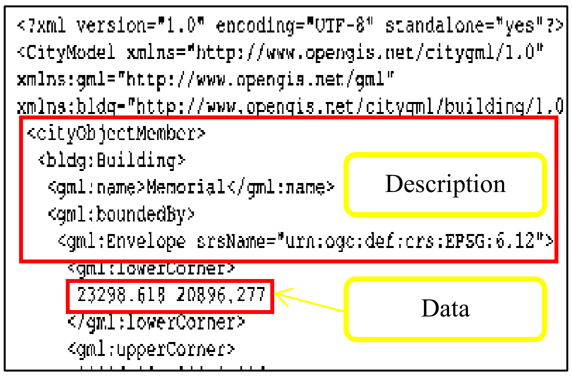
This paper will discuss about the evolution of Geography Markup Language (GML) file compression model.GML is a type of XML files normally used to store spatial data from database. However due to the huge size, processing and transferring this type of file will cost performance and storage issue. Throughout the years several GML file compression model has been developed to help in addressing this problem. Four GML file compression model which are GPress, Delta Spatial Data Compression and Extrapolation Model, GMill, and GPress++ has been selected to be discussed in this paper. In addition a comparison and the enhancement done in each model will be discussed in here. From the assessment GPress++ compression model has shown a significant file compression rate on synthetic dataset with 77.10% improvement on gzip compressor.

Keywords: Geography Markup Language compression, GPress, Delta compression, GMill, GPress++.

I INTRODUCTION

Geography Markup Language (GML) is a file written in XML (Extensible Markup Language). This file is simply a text file and consists of encoded feature-level geospatial data represented by using an open-source GML (Open Geospatial Consortium document 02-023r4 2003). Since the format is universal it can be easily integrated into heterogeneous platforms and devices. In addition, it was able toreduce the costly conversion processes for different database if it was using unstandardized format(Zhang, 2010).

There are two sections of GML file which are the document description and document's content. Figure 1 below illustrates the example of the two sections mentioned.



```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CityModel xmlns="http://www.opengis.net/citygml/1.0"
xmlns:gml="http://www.opengis.net/gml"
xmlns:bldg="http://www.opengis.net/citygml/building/1.0
 <cityObjectMember>
  <bldg:Building>
   <gml:name>Memorial</gml:name>
   <gml:boundedBy>
    <gml:Envelope srsName="urn:ogc:def:crs:EPSG:6.12">
     <gml:lowerCorner>
      23298.618 20896.277
     </gml:lowerCorner>
     <gml:upperCorner>
```

Figure 1. Example of GML file.

Usually the size of the GML files tends to be huge especially files that containsa large number of features. Due to this issue, the network and processing overhead associated with GML makes it unproductive for processing and storage performances (Zhang, 2010).

Researchers have come out with a lot of approaches to reduce GML file size. This paper will continue to discuss the methods proposed in order to reduce the size of GML files.

II GML COMPRESSORS

Several GML compressors have been developed from time to time. GML compressors have been used in order to reduce the size of GML documents. GML compressors that will be discussed here is GPress, delta compression and extrapolation technique, GMill, and GPress++.

A. **Float Point Data Compressor (GPress)**

One of the early models for GML compressor is GPress done by Guan & Zhou (2007). This model later has been adopted by many researchers. Based on the researcher's finding, GPress is the first GML compressor developed.

Basically GML file is using text-based language and the text data is consists of string characters. Since it is in string format, each of the digits in the coordinate will require one byte. Therefore for a coordinate data 10 or even 20 bytes is allocated for storage. Different with XML file in which in binary each single float data is using 4 bytes while double float type is requiring 8 bytes. Because of this situation the size of GML document is normally larger than the XML file.

GPress approach to address this issue is to reduce the precision of the coordinate values. So instead of using double-precision float-point, single-precision is proposed to be used. As an example given coordinate '-101.91555786132812' will be decreased into '-101.915558'. In this way a big amount of space can be saved since we will be needing less bit to represent the differences rather than using full coordinates representation.

GPress method has three compression principles which are:

- Separating document structures from data items. In GPress the structure of GML is extracted and uses a set of integers to keep the positions of tags and data items. Currently, digits 0 to 8 are used to encode the positions of special tokens in the structure.
- Grouping data items based on their semantics. This second principle is to class data items with similar properties together in the same containers according to their semantics.
- Compressing spatial data separately from non-spatial data. Specific containers are allocated to store spatial data and a specific float data delta compression technique is used for the spatial data.

Figure 2 below shows the result from the experiment conducted by Guan & Zhou (2007) to test GPress compression rate. GPress shows a better compression rate than XMill with nearly 20% compression rate.

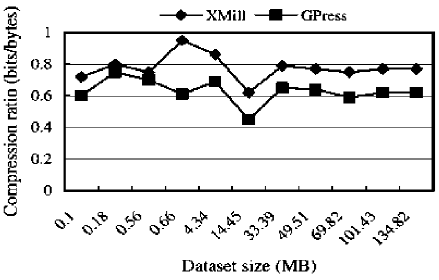
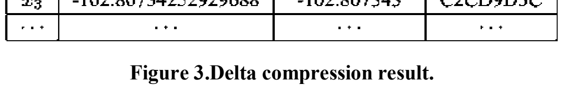


Figure 2.GPress compression rate.

B. **Delta Spatial Data Compression and Extrapolation Technique**

GML files comprise large amount of duplicate data. For this reason, a model proposed by Li, Imaizumi, & Guan (2008) addressed the data redundancy issue. In this model they are using delta compression and extrapolation techniques for GML spatial data compression. Delta compression technique is actually reused from GPress model described above. The improvement done in this model is using extrapolation technique after applying delta compression technique.

Delta compression techniqueis focusing on reducing the long length tags and long sequence number of coordinate's data. As shown is Figure 1, the coordinate is written in float data type thus consuming memory space. Delta compression algorithm is being used to decrease the precision of the coordinate value from double precision number to single precision. Even though this method will produce lossy compression, the result is acceptable since no real world data is this accurate and some systems won't require a full precision data. Figure 3 below shows the coordinate value converted with this algorithm.



		precision decrease	hexadecimal
x_1	-102.91555786132812	-102.915558	C2CDD4C4
x_2	-102.89140319824219	-102.891403	C2CDC866
x_3	-102.80734252929688	-102.807343	C2CD9D5C
...	...	...	...

Figure 3.Delta compression result.

Extrapolation algorithm is designed to compress the coordinate float data. Langrange polynomial is used to forecast the predicted value and the original value. In case the prediction is close to the original value, the difference will be encoded with just a few bits. As an example for

000000101 digits, it will be converted into 101 to truncate the meaningless bits. In this way it will save a big amount of space since we need fewer bits to represent the differences rather than using full values.

Table 1 below displays the simulation result from this experiment done by Li, Imaizumi, & Guan (2008). As we can see, this model achieved a better compression rate compared to XMill compressor in which has the best average compression performance. At the end it able to obtain 22.46% improved rate. In average this technique use more time than XMill due to additional time needed to separate spatial data from non-spatial data.

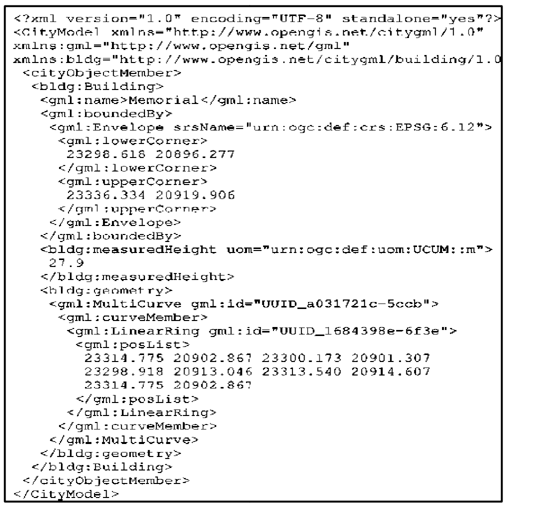
Table 1. Simulation result.

File name	Size	XMill compression	XMill time cost (s)	Compression rate (bits/byte)	Delta compression without precision decrease	Compression rate (bits/byte)	Time cost (s)	Improved Rate (%)
MEX-LAKE	10.3K	1.92K	0.010	1.49	1.68K	1.3	0.010	12.75
MEXROADS	176K	17.7K	0.030	0.8	15.4K	0.7	0.040	12.5
ROADS	1.46M	222K	0.300	1.22	188K	1.03	0.320	15.57
USA-ADI	2.81M	384K	0.510	1.09	294K	0.84	0.570	22.94
ARC	3.86M	417K	0.721	0.86	287K	0.59	0.730	31.4
ROAD3	4.74M	418K	0.721	0.71	292K	0.49	0.741	30.99
WATER4	5.58M	603K	0.971	0.86	427K	0.61	1.001	29.07
WATER2	7.30M	843K	1.371	0.92	616K	0.68	1.482	26.09
MIFBOU2P	8.21M	669K	1.271	0.65	465K	0.45	1.462	30.77
WATER3	11.6M	1.39M	2.243	0.96	1.03M	0.71	2.393	26.04
USA-COUNTRIES	16.4M	1.50M	2.514	0.73	1.25M	0.61	2.673	16.44
COUNTRY	23.9M	2.59M	3.795	0.87	2.22M	0.74	4.806	14.94
Average								22.46

C. **Online Semantic Clustering (GMill)**

The third model will be discussed is compression method based on online semantic clustering. This model is inheriting the delta compression technique in GPress. The difference is that this new model employs semantic similarity of data to assist compression(Wei & Guan, 2010).

Semantic similarity is a benchmark to measure the likeliness of a set of terms on their meanings. In this method digit 1 is used to indicate an extremely high similarity while 0 signifying little or none. The semantic similarity is exploited from these two aspects which are tags and texts.



```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CityModel xmlns="http://www.opengis.net/citygml/1.0"
xmlns:gml="http://www.opengis.net/gml"
xmlns:bldg="http://www.opengis.net/citygml/building/1.0
 <cityObjectMember>
  <bldg:Building>
   <gml:name>Memorial</gml:name>
   <gml:boundedBy>
    <gml:Envelope srsName="urn:ogc:def:crs:EPSG:6.12">
     <gml:lowerCorner>
      23298.618 20896.277
     </gml:lowerCorner>
     <gml:upperCorner>
      23336.334 20919.906
     </gml:upperCorner>
    </gml:Envelope>
   </gml:boundedBy>
   <bldg:measuredHeight uom="urn:ogc:def:uom:UCUM::m">
    27.9
   </bldg:measuredHeight>
   <bldg:geometry>
    <gml:MultiCurve gml:id="UUID_a031721c-5ccb">
     <gml:curveMember>
      <gml:LinearRing gml:id="UUID_1684398e-6f3e">
       <gml:posList>
        23314.775 20902.867 23300.173 20901.307
        23298.918 20913.046 23313.540 20914.607
        23314.775 20902.867
       </gml:posList>
      </gml:LinearRing>
     </gml:curveMember>
    </gml:MultiCurve>
   </bldg:geometry>
  </bldg:Building>
 </cityObjectMember>
</CityModel>
```

Figure 4. GML file example of the features of a building object description.

Figure 4 above displays the GML file describing the features of a building object. From this example, we can see that

- The attribute value "UUID_a031721c-5ccb" of the element "MultiCurve" and the attribute value "UUID_1684398e-6f3e" of the element "gml:LinearRing" are both using the same tags named "gml:id".

- The element content "urn:ogc:def:crs:EPSG:6.12" are linked with the tag name "srsName", and element content "urn:ogc:def:uom:UCUM:: m" are linked with the tag "uom" are actually expressed by similar texts which are "urn:ogc: def:".

From this observation we can conclude that some of the tags and texts are identical. By exploiting these two data characteristics, a GML compression approach by using online semantic clustering or GMill is proposed by Wei & Guan (2010).

Table 2: GMill compression rate.

Datasets	gzip	XMill	GPress	GMill-offdelta	GMill
CityGML	0.5895	0.5071	0.5071	0.5033	**0.4863**
OSMasterMap	0.4807	0.3443	0.3444	0.3430	**0.2690**
Synthetic	0.8408	0.6502	0.4539	0.6485	**0.4522**
Average	0.6370	0.5005	0.4351	0.4983	**0.4025**

Table 2 above shows the compression rate result based on the experiment implemented. From this result GMill outperforms other compressors with 0.4025 average compression rates. However GMill still falls behind the other compressors in term of compression and decompression speed as shown in Table 3 and Table 4 below.This is probably because gzip compressor ignores structures. It deals with structures and data items as normal texts without any complex analyzing and encoding. Other than that, GMill require much time in computing the semantic similarity of data items compared to other compressors in which exploit semantic similarity of data items reflected by paths or tag names.

Table 3. Compression speed.

Datasets	gzip	XMill	GPress	GMill-offdelta	GMill
CityGML	**17.94**	9.79	5.00	3.27	1.57
OSMasterMap	**12.82**	7.18	3.87	2.28	1.61
Synthetic	**1.13**	0.47	0.30	0.19	0.16
Average	**10.63**	5.81	3.05	1.91	1.11

Table 4.Decompression speed.

Datasets	gzip	XMill	GPress	GMill-offdelta	GMill
CityGML	**23.67**	21.28	16.95	13.92	2.40
OSMasterMap	**20.30**	16.19	11.63	9.75	3.48
Synthetic	**2.02**	1.16	0.32	0.89	0.27
Average	**15.33**	12.88	9.63	8.18	2.05

D. **Spatial Proximity (GPress++)**

Another enhancement on GPress compressor described before is spatial proximity based compression method by Wei & Guan (2013) called GPress++.

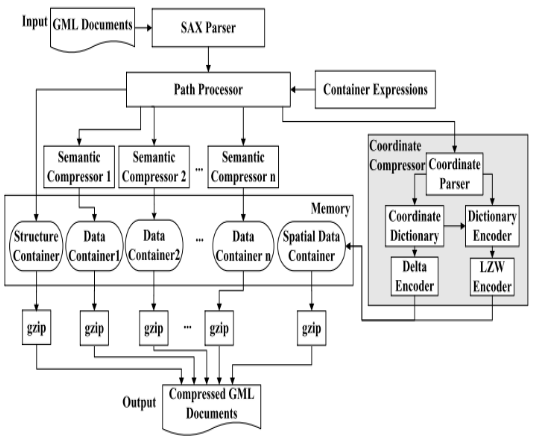


Figure 5. GPress++ Architecture.

Figure 5 shows the architecture of GPress++. At start it will parse the GML file first. After that the output in which is the parsed structural tags are encoded as integers. It will be saved in the structure container. The parsed data is separated into different groups following the path of the original element. Different semantic compressors will compress the data in different groups and directed to the corresponding data containers. Later, memory will store all the containers. Gzip compressor will compress all of the containers and write to the output compressed GML file at the end. Spatial data item s is grouped and Coordinate Compressor will compress it. After that it will forward the compressed file to the spatial data container.

The obvious enhancement of GPress done by GPress++ is the Coordinate Compressor. As we can see, it contains five major modules which are Coordinate Parser, Coordinate Dictionary, Dictionary Encoder, Delta Encoder, and the LZW Encoder. Figure 6, 7, and 8 below shows the result of the experiment conducted.

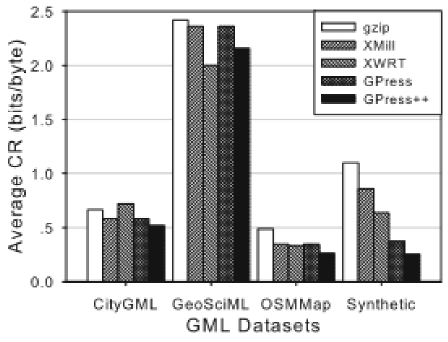


Figure 6.Average Compression Rate.

Figure 6 above shows the average compression rate results of the five compared compressors. As we can see GPress++ outperforms the other compressors except for the GeoSciML dataset.

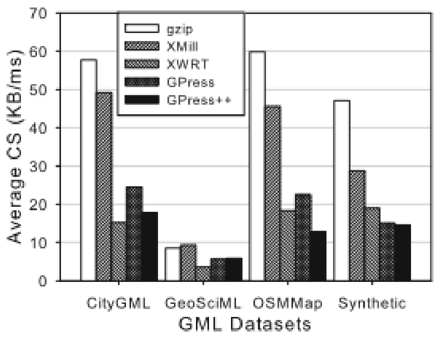


Figure 7.Average Compression Speed.

Figure 7 above shows the average compression speed result. GPress+ performs moderately but on GeoSciML it achieves 63.5% improvement on XWRT.

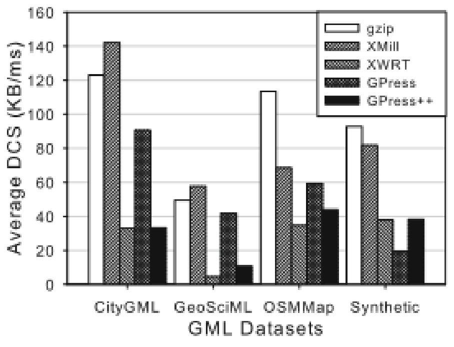


Figure 8.Average Decompression Speed.

Figure 8 above shows the average decompression speed result. GPress++ performs moderately but achieves a 95.2% improvement in GPress on Synthetic dataset.

III COMPARISON

Table 5 below displays the comparison between the compression models mentioned above. This comparison is based on the experiment result described in the research papers. The outcome of these experiments might differ depending on the compressors selected to be compared, dataset, and dataset size.

Table 5. GML Compressors comparison.

No.	Compression Model	Experiment Results							
		Author (s)	Compressors	Dataset	Dataset Size	Compression rate (bits/byte)	Compression speed (KB/ms)	Decompression speed (KB/ms)	Improved rate (%)
1	GPress	1) Jihong	1) XMill 2) GPress	Real geographic features of USA,	10KB to 135MB	Below 0.8			~20

		Guan 2) Shuigen g Zhou		Canada, and Mexico, including cities, highways, rivers, lakes, states etc.					
2	Delta Compression and Extrapolation Technique	1) Yuzhen Li 2) Takashi Imaizu mi 3) Shiro Sakata 4) HirooSe kiya 5) Jihong Guan	1) XMill 2) Delta compression without precision decrease.	Real geographic features of USA, Canada, and Mexico, including cities, highways, rivers, lakes, states etc.	10.30KB to 23.9MB	0.7292			22.46
3	Online Semantic Clustering (GMill)	1) Qingtin g Wei 2) Jihong Guan	1) gzip 2) XMill 3) GPress 4) GMill-offdelta 5) GMill	Select randomly 10 GML documents from: 1) CityGML 2) OSMasterMap 3) Synthetic	543KB to 84MB	0.4025	10.63	15.33	gzip→ 36.8 XMill→ 19.58 GPress → 19.20 Gmill-offdelta → 23.80
4	Spatial Proximity (GPress++)	1) Qingitn g Wei 2) Jihong Guan	1) gzip 2) XMill 3) XWRT 4) GPress 5) GPress++	Select randomly 15 GML documents from: 1) CityGML 2) GeoSciML 3) OSMMap and 5 synthetic documents.	212KB to 84MB. 2.5MB to 1GB (Synthetic)	All compressor average compression rate lies between 0.2 and 2.5.	All compressor compression speed lies between 9.3 and 60.0.	All compressor compression speed lies between 4.6 and 142.2.	gzip→ 77.10 (Syntheti c) GPress → 32.0 (Syntheti c)

IV **DISCUSSION**

Based on Table 5, initially the first known GML compressorwhich is GPress obtained significantly improved rate 20% over XMill. The second model performed slightly better with 22.46% improved rate. This model inherits GPress compression technique but enhanced by using extrapolation technique. The third model which is GMill has a varied result due to comparison against multiple compressors. However it able to achieved the best result which is 36.8% against gzip compressor. GMill still outperformsXMill with 19.58% and its predecessor GPress with 19.20%.

The second generation of GPress which is GPress++ is able to obtain average compression rate between 0.2 and 2.5. This model outperforms other compressors except performing worse than XWRT compressor on GeoSciML dataset. GPress++ achieves 77.10%

improvement on gzip compressor and 32% improvement on GPress with synthetic dataset. Synthetic dataset are actually transformed from the same Oracle Spatial sample fie. This file contains information about road features of USA and Canada, and their size increased from 2.5MB to 1GB.

Regarding the average compression speed GPress++ lies between 9.3 and 60.0. GPress++ performs moderately in average compression speed but able to achieve 63.5% improvement on XWRT with GeoSciML dataset. On average decompression speed, the result lies between 4.6 and 142.2. GPress++ also performs moderately in average decompression speed. However it still obtained 95.2% improvement in GPress on Synthetic dataset.

V CONCLUSION

GML files with high precision spatial data have a huge number of storage sizes thus affecting transition and parsing performance. Based on the discussion we can see several researches has been conducted and experimented to address this issue. Even though results have shown tremendous improvement from time to time in term of compression rate, there is still a lot of room for improvement especially for compression and decompression speed. An alternative to improve the GPress++ model is to adopt DELTA++ encoding technique (Samteladze& Christensen, 2014). This encoding technique has been enhanced from delta encoding used in the first known GML compressor GPress in which has been inherited by all of the other models described above.

ACKNOWLEDGEMENT

This paper is funded by “A New Holistic Service Oriented Model and Sustainable Flood Management: From Monitoring to Emergency Response” under Long Term Research Grant Scheme (Grant ISO No. 12491) from Ministry of Education Malaysia.

REFERENCES

Guan, J., & Zhou, S. (2007). GPress: Towards Effective GML Documents Compresssion. *2007 IEEE 23rd International Conference on Data Engineering*, 1473–1474. doi:10.1109/ICDE.2007.369039

Li, Y., Imaizumi, T., & Guan, J. (2008). Spatial Data Compression Techniques for GML. *2008 Japan-China Joint Workshop on Frontier of Computer Science and Technology*, 79–84. doi:10.1109/FCST.2008.8

Samteladze, N., & Christensen, K. (2014). Feature : Cellular Traffic Reduction DELTA ++ : Reducing the Size of Android Application Updates. *IEEE Internet Computing*, 50-57

Wei, Q., & Guan, J. (2010). A GML compression approach based on on-line semantic clustering. *2010 18th International Conference on Geoinformatics*, 1–7. doi:10.1109/GEOINFORMATICS.2010.5567910

Wei, Q., & Guan, J. (2013). A Spatial Proximity Based Compression Method for GML Documents, 465–471.

Zhang, C. (2010). Develop a Spatial Decision Support System based on Service-Oriented Architecture, (January).