On the Compression of Geography Markup Language [⋆]

Nieves R. Brisaboa[∗], Antonio Fariña[∗], Miguel Luaces[∗], José R. Rios Viqueira[†] and José R. Paramá[∗]

[∗]Database Lab., Univ. da Coruña,
Facultade de Informática, Campus de Elviña s/n,
15071 A Coruña, Spain.
[†] Dept. Electronics and Computer Science
Univ. Santiago de Compostela, Fac. de Fisica, Campus Universitario Sur,
15782 Santiago de Compostela, Spain.
{brisaboa, fari, luaces, parama}@udc.es and joserios@usc.es

Abstract. The Geography Markup Language (GML) is a standard XML-based language that enables the representation and easy interchange of geographic data between Geographic Information Systems (GIS).
In this paper, we explored the compressibility of GML performing some empirical experiments over real GML corpus using a wide set of well-known compressors. In particular, the main characteristics of GML are first described. Next, it is shown how these characteristics can be exploited to achieve a better compression rate on GML files. We use these ideas to design a specific parser and a strategy to compress the representation of geographic objects (their coordinates in a map). Finally, to check the correctness of our hypothesis, the same GML files are compressed by applying the new parser and coordinates encoding strategy.

1 Introduction

In the last years, the technology underlying Spatial Databases and Geographic Information Systems (GIS) has undergone a great development. As a consequence, public administrations and governments are increasingly demanding tools and applications based on Geographic Information Systems (GIS) technology. This technology uses spatial extensions for object-relational databases to store the geometry of geographical objects. That is, these applications not only use standard alphanumeric data, but also geographic data to represent both the shape and the situation in the territory of different geographic objects (plots, roads, buildings, rivers, etc.).

The use of many different commercial off-the-shelf GIS tools, supporting different geographic data formats, makes the interchange of data among different systems difficult. To overcome these inter-operability problems, a standardization effort has been undertaken by the Open Geospatial Consortium (OGC). An

[⋆] Partially supported by MCyT (PGE and FEDER) grant(TIC2003-06593) and Xunta de Galicia grant(PGIDIT05SIN10502PR).

important result of such an effort is the definition of the Geography Markup Language (GML), which is an XML language that enables the representation of objects with both alphanumeric and geographic attributes of any kind.

The last version of this language has recently become a draft of the ISO technical committee 211, which is in charge of standardization in the field of digital geographic information.

Geographic datasets usually contain many geographic objects, each of them defined in terms of a long list of floating point coordinates. Therefore, representing such datasets with GML usually leads to huge text files and consequently to serious efficiency problems in their storage and transmission over the Internet.

To the best of our knowledge, no compression techniques have ever been used to compress GML. Probably this is due to the fact that the systematic use of GML is new, in fact the ISO committee is still standardizing GML. On the other hand, a good part of the GIS community does not have a computer science background.

In this paper, the application of various compression techniques to GML files has been investigated. In particular, in Section 2 the main characteristics GIS and GML are first described. Next, in Section 3, it is shown how these characteristics can be exploited to achieve a better compression rate on GML files and, also in this section, we present strategies to parser and to process coordinates in order to obtain a better compression by taking advantage of those features. Finally, in Section 4, the compressibility of GML is tested using real GML corpus and a wide set of well known compressors. We first use all the compressors directly over the GML files to have a baseline to compare with, and then, we explore the utility of our strategies to improve the compression ratio.

There are several well-known classic compression techniques such as Huffman [18] or Ziv-Lempel [27]. However, the widespread of the web caused the development of a wide range of new compression techniques designed to save storage space and/or transmission time.

Some of these compression methods [22, 21, 12, 14, 13] are *statistical* (also known as "zero-order substitution" methods). Statistical compression techniques split the original text into *symbols* and each symbol is represented (in the compressed text) by a unique *codeword*. Compression is achieved by assigning shorter codewords to more frequent symbols. These techniques need to compute the frequency of each original symbol and then a coding scheme is used to assign a codeword to each symbol.

Other compression methods [17, 27] are based on the use of a *dictionary*. These methods substitute the occurrence of strings in the text by pointers (all of them with the same length) to the correct entry in the dictionary. The longer the strings, the better the compression. These methods take advantage of the *co-occurrence* of characters or words because it permits, in general, longer strings and shorter dictionaries.

Both kinds of compression methods can be either static (vocabulary/dictionary fixed in advance), semistatic or dynamic. Semistatic statistical compression methods are also known as two pass methods [22, 21, 12, 14] since they perform two passes over the original text. In the first pass, they compute the frequency of each symbol, then the coding schema of each method assigns a codeword to each source symbol in the vocabulary and finally, in the second pass, each input symbol in the original text is substituted by its corresponding codeword. Something similar happens with semistatic dictionary-based methods [17].

Classic statistical compression methods used characters as input symbols. However, Moffat in [20] proposed the use of words instead of characters as the symbols to be compressed. By using words instead of characters compression ratios improve drastically, because the distribution of words is much more biased than that of characters.

There are still other well-known compression methods that follow different strategies. For example, arithmetic compressors [16] represent the whole text with a single number depending on the probability of the words in the text. PPM [11] is a statistical compressor that uses arithmetic encoding. Finally, since XML has been proposed as a standard to represent documents some research has been oriented to explode the structure of the documents to get a better compression. SCM-PPM [10] is an example of this kind of compressors.

2 Geographic Information Systems: Basic Concepts

For the purposes of the present paper, *geographic or spatial data* is any kind of data with a reference to *Geographic Space*, i.e., to the Earth surface. Two main categories of *spatial data* have been identified in the GIS literature [23, 24], *spatial objects* and *spatial fields*. A *spatial object* is described by a collection of properties of conventional data types (integer, real, string, etc.) and a collection of properties of spatial data types (point, line, surface), the latter defining its position and shape in *Geographic Space*. Examples of spatial objects are cities, rivers, roads etc.

A *spatial field* is a mapping from the positions of a subset of *Geographic Space* to the domain of some conventional property. A *spatial field* may be either discrete or continuous. In a *discrete field*, a single conventional value is assigned to each different element of a finite collection of either points or lines or surfaces. Examples of discrete spatial fields are the soil type, vegetation type, etc. In the case of *continuous fields* each point of an infinite subsect of *Geographic Space* is mapped to a value of a conventional domain. Examples of continuous spatial fields are the temperature, elevation above see level, etc.

A Spatial Data Infrastructure (SDI) includes a collection of spatial data sources and services. The services are useful to discover, access and process the data in the data sources integrated in the SDI. To guarantee the interoperability between the services of a SDI, the definition of standards was something mandatory. The Open Geospatial Consortium (OGC) [9] has proposed standard specifications for the interfaces of web services that enable a uniform access to heterogeneous collections of spatial data sources. In particular, a Web Feature Service (WFS) may be used to access repositories of *spatial objects* (*Features with geometry* in OGC terminology) in the web. Similarly, a Web Coverage Service (WCS) may be used to access repositories of *spatial fields* (*coverages* in OGC terminology). Finally, a Web Map Service (WMS) may also be used to generate maps (images in formats such as JPG, PNG, SVG, etc.) by assigning visualization styles (colors, line widths, etc.) to the data retrieved from WFSs and WCSs. Commercial and open source tools already exist that implement the standard WFS [1, 4, 8, 6, 7], WCS [1, 3] and WMS [1, 8, 6, 7] interfaces above. To represent the data retrieved by the WFS, the OGC has also defined an XML language called Geography Markup Language (GML). Versions 1.0, 2.0 and 2.1 of GML support the representation of both conventional and spatial properties of *spatial objects*. Among other pieces of functionality, support for the representation of *spatial fields* has been added to GML in versions 3.0 and 3.1. Therefore, GML can now also be used to represent data retrieved by a WCS. Version 3.1 of GML is currently a draft of the ISO technical committee 211, which is in charge of the standardization in the area of digital geographic information. Finally, it is remarked that most of the GML currently used is coded with version 2.1, therefore the remainder of this paper is restricted to this version. According to this, the following subsection gives a brief description of the representation of spatial objects in GML 2.1.

It is interesting to point out that all these standards are in full use since its definition because they were fully accepted by the GIS community. In Europe, the INSPIRE (INfrastructure for SPatial InfoRmation in Europe) [5] initiative of the European Commission intends to trigger the creation of an European SDI, which integrates national, regional and local SDIs of the member states. To establish a legal framework for the creation and operation of such and SDI, a directive of the European Parliament and of the Council has also been proposed by the Commission. Such a directive will force public administrations at national, regional and local level to follow the INSPIRE principles making mandatory the use of the described standards including GML. Therefore in the near future more and more GML files will be exchanged among spatial databases, at less in Europe. As a consequence, the interest of developing techniques for GML compression is clear.

2.1 Representing Spatial Objects in GML

To model *Geographic space*, a *Coordinate Reference System* (CRS) [19] is required, which assigns a tuple of numeric coordinates to each of its positions. Various types of CRSs have been used since the first cartographic representations of the Earth surface. As an example, in a *geographic CRS* positions are expressed in terms of latitude and longitude coordinates. Notice however that because of the curvature of the Earth surface, in order to display its positions in a 2-D flat surface (screen, paper sheet, etc.), a projection is needed. Thus, in a *projected CRS*, positions are expressed in terms of 2-D cartesian coordinates, and a projection is used to map each such position to the relevant position in *Geographic Space*. The *Universal Transverse Mercator* (UTM) is a well-known example of such a projection. The generation of UTM coordinates for each position in the Earth surface is next informally illustrated. First, the shape of the

Earth is approximated by an ellipsoid and each of its positions projected to a horizontal cylinder that surrounds the spheroid. Then, the cylinder is unrolled from north to south and the resulting flat surface partitioned into 60 vertical zones, each of them of 6 degrees wide. The equator splits each such zone into two subzones, north and south. A different origin of coordinates is assigned to each of the above subzones. Both north and south subzones have their origin X coordinate 500,000 meters west from the central meridian of the zone. However, the origin Y coordinate of north and south subzones is placed, respectively, at the equator and 10,000,000 meters south from the equator. These origins guarantee that both X and Y coordinates of positions are always positive distances.

In order to represent in a computer the infinite number of points contained in a spatial object, discrete finite representations had to be developed. Two main types of such discrete representations can be found in the literature [23, 24], namely *vector* and *raster representations*. Roughly speaking, in a *vector representation*, the coordinates of the CRS are approximated by either integer or floating point numbers. A *point* is represented by a pair of (x, y) coordinates of the CRS. *Lines* are approximated by sequences of line segments, each of them represented by its two end-points. A *surface* is represented by the vector approximation of its boundary, which is in the general case composed of an exterior boundary line and the boundary of a collection of holes. In a *Raster representation*, *Geographic Space* is partitioned into a collection of disjoint surfaces, called cells, of the same size and shape (usually squared). A *spatial object* is approximated by a set of such cells. Finally, it is well-known that a *vector representation* achieves a more precise representation of *spatial objects* and *discrete spatial fields* with a low storage cost, whereas, *raster representations* are better suited for the representation of *continuous fields* [24]. Therefore, the representation used by GML for spatial objects is vector-based.

To illustrate the use of GML in a realistic example consider the schema shown in Figure 1. It enables the representation of collections of municipalities in GML.

```
<?xml version="1.0"?>
<xsd:schema targetNamespace="http://www.core06.mx/Ex"
    xmlns="http://www.w3.org/2001/XMLSchema"
    xmlns:ex="http://www.core06.mx/Ex"
    xmlns:gml="http://www.opengis.net/gml" elementFormDefault="qualified">
 <import namespace="http://www.opengis.net/gml" schemaLocation="feature.xsd"/>
 <import namespace="http://www.w3.org/1999/xlink" schemaLocation="xlink.xsd"/>
 <element  name="Example"  type="AbstractFeatureCollectionType"
    substitutionGroup="gml:_FeatureCollection"/>
 <element  name="Municipality"  type="ex:MunicipalityType"
    substitutionGroup="gml:_Feature"/>
 <complexType name="MunicipalityType"> <complexContent>
   <extension base="gml:AbstractFeatureType"><sequence>
      <element name="id" type="long"/> <element name="name" type="string"/>
      <element name="population" type="float"/>
      <element name="geo" type="gml:PolygonPropertyType"/>
   </sequence> </extension>
 </complexContent> </complexType>
</schema>
```

Fig. 1. XML Schema of a GML 2.1 Document.

The *import* element in line (6) specifies the location of the file "feature.xsd" that contains the schema for the definition of GML features (it is reminded that a feature with geometry is the term used by the OGC to denote a spatial object). Next, two elements are defined in the example schema. First an "Example" element is declared as a subtype of the GML type *AbstractFeatureCollectionType*, which enables the representation of collections of spatial objects. It is remarked here that the OGC defines that a collection of features is also a feature. Next, a "Municipality" element is declared. The definition of its type "MunicipalityType" follows in lines (12-18). It is noticed that "MunicipalityType" is defined as a subtype of the GML *AbstractFeatureType*, which is also defined in "feature.xsd" and enables the representation of spatial objects. Besides the general purpose properties already defined in type *AbstractFeatureType*, "MunicipalityType" contains also application specific conventional properties of each municipality. These are the "id" and "name" of the municipality. Finally, a spatial property "geo" of spatial data type polygon is also included in the "MunicipalityType".

```
( 1)<?xml version="1.0"?>
( 2)<Example xmlns="http://www.opengis.net/gml"
( 3)         xmlns:ex="http://www.core06.mx/Ex">
( 4)<boundedBy><Box><coord><X>308787.49</X><Y>4744080.86</Y></coord>
( 5)  <coord><X>315101.36</X><Y>4748098.29</Y></coord></Box></boundedBy>
( 6)<featureMember> <ex:Municipality>
( 7)  <ex:id>1</ex:id><ex:name>Mazaricos</ex:name>
( 8)  <ex:geo><Polygon srsName="EPSG:23031"><outerBoundaryIs>
( 9)   <LinearRing><coordinates>
(10)      309440.29,4744357.47 309038.81,4744668.04 308787.49,4745676.36
(11)      310118.86,4746562.93 310363.24,4747445.11 311109.89,4747560.09
(12)      312741.84,4748098.29 313455.53,4747914.19 309440.29,4744357.47
(13)   </coordinates></LinearRing></outerBoundaryIs></Polygon></ex:geo>
(14)</ex:Municipality> </featureMember>
(15)<featureMember> <ex:Municipality>
(16)  <ex:id>2</ex:id><ex:name>Oroso</ex:name>
(17)  <ex:geo><Polygon srsName="EPSG:4230"><outerBoundaryIs>
(18)   <LinearRing><coordinates>
(19)      -99.0249938,56.6880477 -99.0258258,56.687203 -99.0255803,56.6863047
(20)      -99.0248452,56.6854873 -99.0241594,56.685209 -99.023588,56.6851012
(21)      -99.0223964,56.6851823 -99.018169,56.6861353 -99.0249938,56.6880477
(22)   </coordinates></LinearRing></outerBoundaryIs></Polygon></ex:geo>
(23) </ex:Municipality> </featureMember>
(24)</Example>
```

Fig. 2. Example of a GML 2.1 Document.

Based on the GML schema described above, a GML document containing a collection of two municipalities is shown in Figure 2. First, the minimum bounding rectangle that contains all the spatial objects in the represented collection is declared in the *BoundedBy* element in lines (4-5). It is noticed that the coordinates of such a rectangle are given in the "EPSG:23029" CRS, name given by the European Petroleum Survey Group to the UTM Zone 31 north. The *BoundedBy* element is mandatory for feature collections. Next the first member municipality of the collection is represented. Line (7) represent the conventional properties of the municipality ("id" and "name"). Next, lines (8-13) define the spatial property "geo", whose data type is polygon and whose coordinates are again coded with the UTM CRS. The second municipality of the collection is represented in lines (15-23). Now, the coordinates of the polygon of this municipality are coded in degrees of latitude and longitude.

In the GML example in Figure 2, it can be observed that an important part of the document is occupied by numeric coordinates. Obviously, the number of coordinates depends much on the precision of the represented data. Usually, the percentage of document occupied by coordinates is low in collections of point spatial objects and large in collections of lines and surfaces, reaching in some cases an amount of more than 80% of the file.

3 Compressing GML

GML documents are a special kind of textual documents, therefore we though that compressing them as any other textual document, without considering their specificity, would lead to losing compression ratio. Our hypothesis is that there are some GML features that can be exploited to increase its compressibility. In the next section we show the empirical data that proves such affirmation. In this section, we describe the GML characteristics that we have exploited to improve the compression ratio. Those features are:

1. GML files include many tags and numbers, which represent coordinates. On the other hand, the alphanumeric part represents data extracted from the columns of the non-spatial part of the database. Therefore it can not be considered as a natural language document. Our hypothesis was that the word frequency distribution in GML documents would be very different of those typical in natural language documents.
   We checked this hypothesis and computed the word frequency distribution of GML text and we found that it could be approximated by the Zipf distribution [26], but the Θ parameter of Zipf distribution, that has a value between 1.2 and 1.6 in natural language text, has in GML text the average value of 0.6. Therefore, we thought that the usual word-based compressors used in natural language documents such as [20–22, 14, 12] would not be efficient, as we will prove later.
2. In natural language documents, words can be usually identified by a sequence of alphabetic characters ('a'..'z', 'A'..'Z', '0'..'9') and everything between words is considered a separator. However, GML is cluttered with tags, and each appearance of a specific alphanumeric attribute of the database is marked with the tag corresponding to the name of such attribute. That is, a tag is written before and after the value itself. Tags are always written between '<' and '>'. Furthermore, other characters such as '=' or '_' appear systematically after some words.

From this characteristic, we argued that parsers used in word-based compressors are not adequate, and we designed a new parser taking into account the specific use in GML of tags and other usual symbols. Then we empirically checked the new parser as shown in Section 4.

3. GML is an XML-based language, as a consequence it can have many indentations, produced by long strings of blank spaces. Such spaces are useless, except to make the documents easier to read by humans. In the compression research field, spaces are always respected as any other character in the text, and they can not be removed. However, in GML, it does not make sense to keep all these spaces. In fact, some WFS, such as "Deegree" [1] do not insert spaces in the GML files, while other applications like JUMP [25], introduce a lot of them. Notice that spaces are not useful to split words because information in GML appears in the middle of the appropriate tags that always can be identified because they start and end with '<' and '>' respectively.
   Hence, we decided to study how the spaces (removed or not) affect the different compressors tested, but our hypothesis was that it would be useful to remove the spaces in the compressed text. We present these results in Section 4.
4. Coordinates represent a significative part (that can be even the 95%) of GML files. Such coordinates describe the points of the vectors that conform the geometry of each spatial object. Coordinates representing a spatial object can be a long sequence of numbers. However, some of the digits of these numbers will be equal in every coordinate, since coordinates represent points in a spatial object geometry, using UTM or Longitude or Latitude coordinates systems. Evidently, a point in an object may not be very far from others in the same object, and therefore, the most significative digits are usually the same since all points of a object belong to the same geographic area.
   On the other hand, coordinates rarely appear in two objects at the same time or twice in the same object. Therefore, introducing each coordinate value in the vocabulary as a new entry, and encoding it with a specific code would not produce compression. Word-based statistical compressor do exactly that and therefore we argue that this kind of compressors will obtain very poor compression ratios.
   To deal with this characteristic we decided to design a strategy to process the coordinates in order to improve the results of different compressors. Specifically, we focused our attention in the word-based statistical compressors since we thought that those compressors would be the most affected ones.

3.1 Strategies to Improve GML Compression

The strategies followed to improve GML compression were:

1. A lossless compressor must be able to compress and later decompress a text obtaining an exact representation of the original text, but we considered that in this case the spaces are meaningless and therefore we decided that removing them before the compression would be acceptable in GML. Empirical results show the gain obtained by doing that.

2. We built a new parser adapted to GML. This parser identifies the characters that are useful in GML to split words. Tags are identified as an unit, and repeated attributes ending with specific symbols such us "=" are identified as a single word. Section 4 presents the effect of this new parser.
3. To avoid the large amount of digits repeated in the coordinates, we decided to represent each coordinate as a difference to the previous one. In this way, the more significative numbers, representing the general geographic area where all the spatial points of the object are placed, are not repeated because after the first coordinate all the others are differences with respect to the previous one. This leads to save a big amount of space since we need less digits to represent differences than to represent full coordinates. On the other hand, to reduce the space used to represent numbers (inside coordinates), we decided to use a compact representation which uses only 4 bits for each number. This gives 12 codes that are enough to represent the 10 digits plus the symbols '$+'$ and '$-'$, needed to represent the sign of the difference with the previous coordinate and, at the same time, to identify the beginning of each coordinate.

Summarizing, we preprocess the GML text removing spaces before to start the compression with any compressor. Then, to improve the compression of word-based statistical compressor, we substituted coordinates by its differences and we represent numbers with a compact representation of 4 bits. In addition, to improve the compression of word-based statistical compressors, we designed a specific parser to identify the best words candidates. This parser identifies as a single word each whole set of compacted differences of consecutive coordinates describing the vectorial representation of a spatial object. Therefore this (maybe long) array of bytes, each byte representing two numbers of the differences among coordinates, is introduced as a single entry into the vocabulary and encoded with a byte oriented codeword (that usually has 3 bytes, or 4 if the file is huge). At the end, the vocabulary is compressed using character-based bit oriented Huffman.

4 Empirical Data

The main target of this section is to test the compressibility of GML files. We used some real GML files extracted from the EIEL Geographic Information System accesible in the web [2]. This system include a huge amount of information about the infrastructures of some Spanish municipalities. We extracted GML files form the following tables: Contour lines (CL), Plots (P), Municipalities borders (MB), Municipalities information(MI), Water supply network (WS), Roads (R) and Road Stretches (RS).

We started checking the amount of spaces that are included in the GML files using different applications. Using *Deegree* to generate the GML file about *Municipalities Borders* we obtained a file of 3,394,257 bytes, while *JUMP*, using the same table of the database, produces a GML file of 7,130,665 bytes. Then we compressed these two files obtaining the results shown in Table 1. Since those spaces are meaningless and produce a loss in compression, we decided to remove them (which is equivalent to use *Deegree* to obtain the GML files).

In Table 2, columns 1 and 2 show the name and size of the files without spaces. Column 3 represents the percentage of the file size occupied by coordinates. Notice that the space occupied by coordinates changes drastically depending on the complexity of the shape of the spatial objects included in the GML file.

All files were compressed with different compressors, some were general-purpose compressors, some were text compressors and finally, some were specially designed to compress XML files. As general-purpose compressors, we included the dictionary-based compressor *gzip* [27], one of the most frequently used compressors, *bzip2*, which is based in the Burrows-Wheeler Transform [15], and an *arithmetic* compressor [16] customized to use characters as symbols.

We also used two word-based and byte oriented statistical compressors: Plain Huffman (PH)[21], a Huffman-based compressor and End Tagged Dense Code (ETDC) [14], less efficient but faster and easier to implement, which is a compressor of the Dense family [12, 13].

Finally, we included *SCMPPM* [10] which is an adaption of the well known predictive compressor PPM [11] specifically adapted to compress XML files.

FILE	**size**	gzip	bzip2	scmppm	arith	ETDC	PH	Class. Huff.
JUMP	7,130,665	1,153,291	904,308	874,192	2,638,057	2,540,224	2,455,558	2,694,058
Deegree	3,413,795	996,145	890,498	810,741	1,749,144	2,381,037	2,229,043	1,802,206

Table 1. Compression removing and without removing spaces (sizes in bytes).

FILE	**size**	**coord.**			scm-		**unmodified**		**xml parser**		**Coord. Diff**	
	(kb.)	(%)	gzip	bzip2	ppm	arith	ETDC	PH	ETDC	**PH**	ETDC	PH
CL	4,322	67.90	20.34	15.83	15.35	60.52	38.88	37.68	33.45	33.44	15.38	15.37
MB	3,414	91.74	29.18	26.09	23.75	51.24	69.75	67.14	45.57	45.53	21.11	21.07
P	8,081	9.85	3.92	3.02	2.67	64.89	27.47	27.28	13.75	13.52	11.59	11.36
MI	12,770	95.98	31.73	29.40	27.69	48.83	53.72	52.63	46.46	46.45	23.53	23.52
WS	25,642	58.43	22.39	19.95	18.45	63.78	45.62	44.78	36.96	36.90	24.45	24.39
R	54,736	82.90	34.90	31.92	30.26	56.06	57.19	56.84	41.82	41.76	29.20	29.15
RS	81,332	55.03	24.05	21.22	19.94	65.16	48.16	47.92	35.12	35.05	26.81	26.75

Table 2. Compression of spaceless files using modified parser and coordinate processing.

First we compare all these techniques without any modification, just to compare the regular version of these compressors when they are applied to GML files. Table 2 shows the compression ratios achieved by the compression techniques included in our study (columns 4 to 9). General-purpose compressors gzip and bzip2 obtain good compression ratios, between 20% and 30%, except in the case of the *Plots* file, where the small percentage of the file occupied by coordinates produces a much better compression. It can be observed that text compressors have problems with GML files, because the streams of coordinates are not compressible with these compressors. Finally, the best results are obtained by the SCMPPM, although it does not take special care of the coordinates. Columns 10 to 13 show the results after adapting the parsers of the text compressors ETDC and PH to the GML characteristics. Furthermore, the influence of using the compact representation of the coordinates as differences can be seen by comparing the compression obtained when such preprocessing of the coordinates is done (columns 12 and 13) with the compression obtained when the compression of coordinates follows the standard procedure of the rest of the file (columns 10 and 11).

As it can be seen, when the coordinates are processed, ETDC and PH become closer to the compression ratios of SCMPPM. On the other hand, ETDC and PH are better than other alternatives when the percentage of space occupied by coordinates is significative.

5 Conclusions

The results presented in this paper demonstrate that we should devote attention to the characteristics of GML in order to compress it efficiently. The ideas shown here are only a first approximation to the problem, obviously they need to be improved, specially the efficiency of the compression/decompression process.

On the other hand, GML is automatically produced by software modules (possibly conforming with the WFS standard) and can be automatically read by other software modules (such as those following the WMS standard). We think that it could be convenient to develop applications including the WFS or WMS standards and compressors, to use a compressed version of GML by pipelining the compressor with the web service. That is, the output of a GML source could be compressed by the appropriate module, and before such a compressed file is provided as input to a GML consumer, a decompression module could decompress it to provide the information in plain form.

Another research line that should be undertaken is the possibility of searching patterns directly in the compressed text. This problem has been tackled, in natural language, by several researchers [21, 14, 12]. However, in natural text. GML represents spatial objects, so different applications could take advantage of the possibility of searching directly into the GML files. However, searching inside of GML involves new constraints and characteristics not present in usual text retrieval tasks.

We believe that this work opens a new field with new challenges to the compression research field. The compression of GML files has different constraints and possibilities that are not present in the compression of other kind of files such as text, DNA, images or music. Furthermore, presumably GML applications will attract more demand day after day and then, more different applications could benefit from the use of good compressors.

References

1. Deegree. URL: http://deegree.sourceforge.net/.
2. The EIEL project. URL: http://www.dicoruna.es/webeiel/.
3. ESRI arcGIS server. URL: http://www.esri.com/software/arcgis/arcgisserver/.
4. The GeoServer project. URL: http://geoserver.sourceforge.net/html/.

5. INSPIRE: INfrastructure for SPatial InfoRmation in Europe. URL: http://inspire.jrc.it/home.html.
6. Intergraph geomedia web map. URL: http://imgs.intergraph.com/gmwp/.
7. Mapinfo mapxtreme. URL: http://extranet.mapinfo.com/products/.
8. Mapserver. URL: http://mapserver.gis.umn.edu/.
9. OGC: Open geospatial consortium. URL: http://www.opengeospatial.org/.
10. J. Adiego, G. Navarro, and P. de la Fuente. Scm: Structural contexts model for improving compression in semistructured text databases. In *Proc. SPIRE 2003*, LNCS 2857, pages 153–167. Springer, 2003.
11. T. Bell, J. Cleary, and I. Witten. Data compression using adaptive coding and partial string matching. *IEEE Transactions on Communications*, 32(4):396–402, 1984.
12. Nieves R. Brisaboa, Antonio Fariña, Gonzalo Navarro, and Maria F. Esteller. (s,c)-dense coding: An optimized compression code for natural language text databases. In *Proc. SPIRE 2003*, LNCS 2857, pages 122–136, 2003.
13. Nieves R. Brisaboa, Antonio Fariña, Gonzalo Navarro, and José Paramá. Simple, fast, and efficient natural language adaptive compression. In *Proceedings SPIRE 2004*, LNCS 3246, pages 230–241. Springer, 2004.
14. Nieves R. Brisaboa, Eva L. Iglesias, Gonzalo Navarro, and José R. Paramá. An efficient compression code for text databases. In *25th European Conference on IR Research, ECIR 2003; LNCS 2633*, pages 468–481, Pisa, Italy, 2003.
15. M. Burrows and D. J. Wheeler. A block-sorting lossless data compression algorithm. Technical Report 124, 1994.
16. John Carpinelli, Alistair Moffat, Radford Neal, Wayne Salamonsen, Lang Stuiver, Andrew Turpin, and Ian Witten. Word, character, integer, and bit based compression using arithmetic coding, 1999.
17. Philip Gage. A new algorithm for data compression. *C Users Journal*, 12(2):23–??, February 1994.
18. D. A. Huffman. A method for the construction of minimum-redundancy codes. In *Proc. Inst. Radio Eng.*, pages 1098–1101, September 1952. Published as Proc. Inst. Radio Eng., volume 40, number 9.
19. Snyder J.P. *Map Projections - A Working Manual*. U.S. Geological Survey Professional Paper 1395, United States Goverment Priting Office, 1987.
20. A. Moffat. Word-based text compression. *Software - Practice and Experience*, 19(2):185–198, 1989.
21. Edleno Silva de Moura, Gonzalo Navarro, Nivio Ziviani, and Ricardo Baeza-Yates. Fast and flexible word searching on compressed text. *ACM Transactions on Information Systems*, 18(2):113–139, April 2000.
22. Gonzalo Navarro, Edleno Silva de Moura, M. Neubert, Nivio Ziviani, and Ricardo Baeza-Yates. Adding compression to block addressing inverted indexes. *Information Retrieval*, 3(1):49–77, 2000.
23. Rigaux P., Scholl M., and Voisard A. *Spatial Databases: with application to GIS*. Morgan Kaufmann Publishers, Academic press, 2002.
24. Burrough P.A. and McDonnell R.A. *Principles of Geographical Information Systems*. Oxford University Press, 1998.
25. Inc. Vivid Solutions. The Jump Project. Available at http://www.jump-project.org, 2005.
26. George K. Zipf. *Human Behavior and the Principle of Least Effort*. Addison-Wesley (Reading MA), 1949.
27. Jacob Ziv and Abraham Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23(3):337–343, 1977.