第36卷 第14期 计 算 机 工 程 2010年7月
Vol.36 No.14 Computer Engineering July 2010
·软件技术与数据库· 文章编号：1000—3428(2010)14—0026—03 文献标识码：A 中图分类号：N945

GML空间数据压缩技术研究

朱 华[1]，李 岩[1,2]
(1. 华南师范大学计算机学院，广州 510631；2. 华南师范大学空间信息研究中心，广州 510631)

摘 要：针对GML空间数据冗余较大且存储和传输代价高的问题，提出一种GML空间数据压缩方法，采用VTD-XML解析GML文档，设计将树形结构的GML空间数据转换为2个线性结构数据的GBW变换，利用GZip压缩数据并输出。实验结果表明，该方法优于传统压缩方法，在提高GML空间数据压缩率的同时，并未明显增加压缩及解压缩时间。

关键词：GML空间数据；压缩；GBW变换；VTD-XML解析

Research on Compression Technology of GML-based Spatial Data

ZHU Hua[1], LI Yan[1,2]
(1. Computer Institute, South China Normal University, Guangzhou 510631;
2. Spatial Information Research Center, South China Normal University, Guangzhou 510631)

【Abstract】 Aiming at the high cost of transmitting and preserving for redundancy GML-based spatial data, a compression method is presented. It uses VTD-XML to parse GML document and designs GBW transform which transforms tree structure of GML-based spatial data to two linear structure data. It uses GZip to compress the data and output it. Experimental result indicates it defeats common methods and can improve compression rate of GML-based spatial data efficiency while not needing more compression and de-compression time.

【Key words】 GML-based spatial data; compression; GBW transform; VTD-XML parser

1 概述

许多涉及海量空间数据共享、交换、集成和服务的WebGIS应用系统，它们以空间信息共享的GML数据格式进行传输和存储处理。由于GML空间数据包括大量冗余的、结构化的空间矢量数据，导致它的存储和传输代价都非常高，因此非常有必要对GML空间数据进行压缩研究。

文献[1]提出一种GML空间数据压缩方法，但此方法较复杂。文献[2]针对GML空间数据流的压缩传输做了研究，只适于数据流的压缩传输。文献[3-4]针对XML的压缩分别做了研究，提出一些压缩方法，但这些XML压缩方法应用到GML中效果不理想。分析国内外的XML压缩技术可发现一个共性：将XML文档的模式(结构数据)和数据分离，并且把相同路径的数据在逻辑上邻近存放，再集成了一些现有的通用数据压缩方法，能达到更好的压缩效果。本文利用此特点，提出一种新的GML空间数据压缩方法。

2 相关理论与定义

2.1 GML空间数据模型

GML采用半结构化的树状结构作为数据的组织形式，每个GML文件都是一棵有向树，树结点的组成类型可以分为3类：模式，属性，数据值，数据值包括空间数据和普通数据2种。以下是一个GML空间数据片段示例。

```
<gml:FeatureCollection>
<gml:featureMember>
<gml:GEO_CODE>24010</gml:GEO_CODE>
<gml:Polygon srsName="a">
<gml:LinearRing>616 031.253 7,…</gml:LinearRing>
</gml:Polygon>
</gml:featureMember>
<gml:featureMember>
<gml:GEO_CODE>24010</gml:GEO_CODE>
<gml:Polygon srsName="b">
<gml:LinearRing>627 396.932 4,…</gml:LinearRing>
</gml:Polygon>
</gml:featureMember>
</gml:FeatureCollection>
```

从示例可看出，gml:featureMember和gml:Polygon等为模式，srsName为属性，“a”为普通数据，616 031.253 7,…为空间数据。空间数据通常精度很高，数据量庞大，此处用了省略符代替。

2.2 GBW变换

假设GML树模型为T，它有t个节点、任意高度和形状，相关定义如下：

定义1 GML文档中每个开始标签名为T的一个节点，且在节点前加上符号“<”；标签的属性名为它的子节点，且在子节点前加上符号“@”，属性值为子节点的孩子；所有叶子节点前加上符号“#”。

定义1约定了T中不同类型节点前都加上了唯一的分隔符，它们都是GML规范所保留的字符，可保证在T中不会有其他的字符与其重复，有2个重要作用：GBW变换中字典替换和排序都需要根据这些分隔符来判断$S\alpha$分量中的数据项是否为叶节点；利用$S\alpha$和$S\pi$分量做GBW反变换重构树

基金项目：国家自然科学基金资助项目(60842007)；广东省重点引导基金资助项目“空间信息共享系统集成与典型示范研究”(2004B32501001)

作者简介：朱 华(1983－)，男，硕士研究生，主研方向：图像处理，空间信息技术；李 岩，教授

收稿日期：2009-12-03 **E-mail：**yanli@scnu.edu.cn

模型时需要相应的标记作为区分。

定义 2 对于 T 的每个节点 t，$\alpha[t]$表示节点 t 的标记名称，$\pi[t]$表示从 t 的双亲节点开始到 T 的根节点结束，自上而下地将这条路径的所有节点连接起来的字符串。

定义 3(GBW 变换)

(1)先序遍历 T，计算每个节点 t 的值 $s[t]=<\alpha[t]，\pi[t]>$顺序插入到二维表 S 中。其中，如果 t 是非叶节点，则 $\pi[t]$为空，否则 $\pi[t]$的值为 t 的路径；$\alpha[t]$的值为节点名。

(2)对 S 中的数据建立字典表并做字典替换，然后再根据 $S\pi$ 中数据值将 S 中数据排序。

(3)令 $S\alpha$ 记录所有的 α 分量，$S\pi$ 记录所有的 π 分量。

由于 GML 树模型 T 中的非叶节点存在大量重复出现且冗余，上述 GBW 变换的第(1)步操作会记录很多重复的非叶节点数据。为了达到更好的压缩效果，可以建立一个字典表，用字典码来替换冗余的非叶节点数据。字典替换算法描述如下：

(1)对 $S\alpha$ 分量中的每个非叶节点数据 $\alpha[m]$，从 $S\alpha$ 中的第 1 项开始检查是否还有其他非叶子节点数据 $\alpha[n]$与其相等。

(2)如果不存在 $\alpha[m]=\alpha[n]$，则给 $\pi[m]$赋一个独一无二的简短数值作为字典码，然后做字典替换：将路径分量 $S\pi$ 中所有的 $\alpha[m]$值都替换为对应的字典码。

(3)如果存在 $\alpha[m]=\alpha[n]$，则说明此非叶节点数据在 S 中被重复记录，删除它所在 S 中的那条记录。

经过字典替换后，再根据 $S\pi$ 中数据值将 S 中数据排序。为了达到相同路径的数据逻辑上相邻存放的目的，本文设计的排序算法根据 $S\pi$ 中叶节点数据的路径是否相同来排序，且使得排序后的相同的路径数据之间仍然相对有序，伪代码描述如下：

```
输入 S
输出 排序后的 S
for (i = 1; i<=Sπ.Length; i++ )
{
If(α[i]是叶节点)
        for (j = i+1; j<=Sπ.Length; j++ )
    {
                If(π[j] == π[i] && α[j]是叶节点数据)
                    将 s[j]移动到 s[i]的后面;
i++;
     }
}
```

2.3 GBW 变换分析

GBW 变换的实质是将树形结构的 GML 空间数据转换为 2 个线性结构的数据，它与文献[3-4]中的 XBW 变换主要区别如下：

(1)两者基于的树模型不同。本文定义的 GML 树模型能很好地解决 XBW 变换中树模型不支持字符串和属性这 2 个问题。

(2)两者的排序算法不同。GBW 变换是基于 $S\pi$ 中整条数据是否相同来排序。

(3)GBW 变换中的 $S\pi$ 分量保存的数据为字典码和以字典码表示的路径数据。

(4)GBW 中没有标志符分量 Slast，且添加了字典替换这个步骤，进一步提高压缩率。

上文的 GML 空间数据示例按照本文的定义建立的树模型如图 1 所示，其 GBW 变换过程中字典替换和排序后的 S 如表 1、表 2 所示。

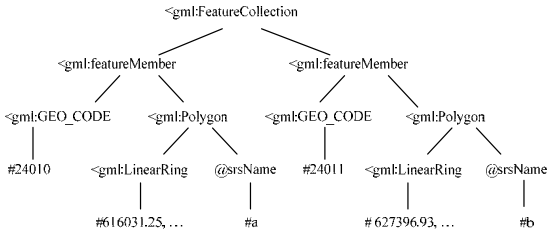

图 1 GML 树模型 T

表 1 字典替换后的 S

$S\alpha$	$S\pi$
<gml:FeatureCollection	1
<gml:featureMember	2
<gml:GEO_CODE	3
#24010	<3<2<1
<gml:Polygon	4
@srsName	5
#a	<5<4<2<1
<gml:LinearRing	6
#616031.25,…	<6<4<2<1
#24011	<3<2<1
#b	<5<4<2<1
#627396.93,…	<6<4<2<1

表 2 排序后的 S

$S\alpha$	$S\pi$
<gml:FeatureCollection	1
<gml:featureMember	2
<gml:GEO_CODE	3
#24010	<3<2<1
#24011	<3<2<1
<gml:Polygon	4
@srsName	5
#a	<5<4<2<1
#b	<3<2<1
<gml:LinearRing	6
#616031.25,…	<6<4<2<1
#627396.93,…	<6<4<2<1

2.4 GBW 反变换

为了解压缩能得到完整正确的 GML 文件，必须设计一个让 GBW 变换后能完整还原的解决方案。根据 GBW 变换的定义，可知 GBW 反变换过程的首要任务是通过字典还原，然后根据路径信息可以重构 GML 文档。字典还原的主要步骤伪代码描述如下：

```
输入 S
输出 还原后的 S
for (i = 1; i<=Sπ.Length; i++ )
{
If(α[i]是非节点数据)
将 π[i]作为字典码，α[i]作为原值添加进字典表 DICT；
 If(α[i]是叶节点数据)
      查找 DICT，将 π[i]中的所有字典码都替换为原值；
}
```

重新构造 GML 文档的过程描述如下：

(1)找出 $S\pi$ 中每个路径数据 $\pi[m]$。

(2)在 $\pi[m]$中，每个用“ < ”隔开的字符串都为一个标签，提取并顺序地生成这些 GML 标签。

(3)将 $\alpha[m]$的数据值写入 GML 文档。

3 压缩框架

本文提出的压缩方法是在传统数据压缩算法的基础上，根据 GML 自身的特点而构造的一个专用于 GML 的高压缩比专用技术，其流程框架如图 2 所示。

图 2 GML 空间数据压缩框架

压缩流程如下：首先，通过 VTD-XML 解析 GML 空间数据，高效的识别出结构数据和内容数据，然后对 GML 数据做 GBW 变换，再对 GBW 变换后的数据采用主流的文本压缩算法 GZip 压缩并输出。其中，VTD-XML 是一种非提取模式的 XML 解析模型，它不仅解决了 DOM 占用内存过大的缺点，并且具有快速的解析与遍历功能，是目前遍历 XML 文件速度最快的解析技术。解压缩正好是压缩的逆过程：首先对压缩的文件采用 GZip 解压缩，然后做 GBW 反变换，即可得到完整的解压缩文件。

4 实验与分析

为了验证本文设计的压缩方法的性能，将其与著名的通用压缩算法 GZip 和 LZMA 在压缩率、压缩时间和解压缩时间 3 个方面作对比实验。实验软硬件环境：Intel(R) Core(TM)2 Duo P8400@2.26 GHz，2 GB RAM，VTD-XML V2.5，GZip V1.2.4，LZMA V4.65。实验数据采用广东省地理空间基础信息，涵盖了河流、公路、铁路、湖泊和行政边界等数十个各种真实 GML 空间数据文档。实验结果如表 3~表 5 所示，这里仅列出 5 组数据，其中，GBWZip 表示本文的压缩算法，LZMA 算法的字典大小设为较小的 3 KB，时间单位为 s，文件大小单位为 KB。可以看出，GBWZip 的压缩率比 GZip 高出平均约 10 个百分点，与 LZMA 基本一致；GBWZip 与 GZip 的压缩和解压缩时间差不多(最多高 2 倍)，而 LZMA 的压缩和解压缩时间是它们的 5 倍~10 倍。这说明 GBWZip 在提高压缩率的同时并不需要耗费大量的时间，具有一定优势。

表 3　压缩率对比

原文件大小/KB	GBWZip/(%)	LZMA/(%)	GZip/(%)
1 244.43	79.06	78.05	62.21
3 890.86	82.28	81.86	74.07
17 351.88	87.37	86.98	75.94
21 054.83	85.21	84.85	71.44
130 877.80	91.46	88.61	85.45

表 4　压缩时间对比

原文件大小/KB	GBWZip/s	LZMA/s	GZip/s
1 244.43	0.28	1.91	0.22
3 890.86	0.74	7.41	0.39
17 351.88	3.45	33.08	1.11
21 054.83	3.69	33.72	1.48
130 877.80	20.20	259.03	9.80

表 5　解压缩时间对比

原文件大小/KB	GBWZip/s	LZMA/s	GZip/s
1 244.43	0.05	0.22	0.03
3 890.86	0.10	0.56	0.06
17 351.88	0.42	2.56	0.34
21 054.83	1.00	3.19	0.36
130 877.80	2.94	10.91	1.61

5 结束语

本文构造了一种将 GML 文档的模式和数据分离的 GBW 变换，并且提出一种 GML 空间数据压缩方法。实验结果表明，该方法具有较高的压缩和解压缩效率。如何进一步提高压缩率并且缩短压缩与解压缩时间是今后研究的方向。

参考文献

[1] Guan Jihong, Zhou Shuigeng. GPress Towards Effective GML Documents Compression[C]//Proc. of the IEEE Int'l Conference on Data Engineering. [S. l.]: IEEE Press, 2007.

[2] 杜成龙, 关佶红, 王　治. GML 空间数据流压缩算法研究[J]. 计算机工程, 2007, 33(1): 98-100.

[3] Paolo F, Fabrizio L, Giovanni M. Structuring Labeled Trees for Optimal Succinctness and Beyond[C]//Proceedings of the 46th IEEE Symposium on Foundations of Computer Science. [S. l.]: IEEE Press, 2005.

[4] 胡智飞, 杨路明, 刘　波. 基于 XBW 变换的 XML 数据压缩查询方法[J]. 计算机工程, 2008, 34(19): 67-69.

编辑　陈　文

(上接第 23 页)

从图 1(c)和图 1(d)的比较中可以看出，图 1(c)得到的运动目标比图 1(d)模糊，而且含有很多噪声像素点。图 1(d)边界光滑，目标边界内外基本无噪声像素。视频帧在时间轴上自动分为 2 个聚类中心，分别是背景类和运动目标类。用 FCM 聚类算法产生的聚类中心落在样本间距离中心，而且容易收敛到局部最优，得不到精确的聚类中心，因此，对运动目标的归属判断力弱，容易出现如图 1(c)所示的误判现象。利用 G-IPSO 算法是基于高斯核函数相似性度量的聚类，聚类目标在概率分布密度最大的区域，符合视频帧数据分布的特点。

每个视频帧含有 704×576=405 504 个像素，每个像素是具有 YUV422 格式的三维数据，连续取 150 帧，则需要处理的数据达到 405 504×3×150=182 476 800 个，这是个非常庞大的数据，如果选择基于梯度迭代的 FCM 算法来对视频背景建模，将耗费巨量时间。随机选取其中含有运动目标的 5 个像素，对此 2 种算法聚类操作所耗费的时间进行比较，如表 1 所示。

表 1　聚类操作耗费时间比较　　ms

聚类算法	Pixel 1	Pixel 2	Pixel 3	Pixel 4	Pixel 5
FCM	13.23	12.45	12.86	13.14	12.27
G-PSO	8.87	8.36	7.25	7.16	8.93
G-IPSO	2.24	2.66	2.42	3.38	3.48

4 结束语

本文提出了基于粒子群优化的高斯核函数聚类算法。实验表明，在不考虑动态背景、遮挡等特殊情况下，该算法在机器视觉领域具有广泛的应用空间。进一步的研究将考虑改进该算法应用于室外的视频背景模型的聚类。

参考文献

[1] Jain A K, Murty M N, Flynn P J. Data Clustering: A Survey[J]. ACM Computer Survey, 1999, 3l(3): 264-323.

[2] 唐贤伦, 庄　陵, 李银国, 等. 基于粒子群优化和模糊 C 均值聚类的入侵检测[J]. 计算机工程, 2008, 34(4): 13-15.

[3] Spagnolo P, Orazio T, Leo M, et al. Moving Object Segmentation by Background Subtraction and Temporal Analysis[J]. Image and Vision Computing, 2006, 24(5): 4l1-423.

[4] Yu Jin, Qian Feng, Qi Rongbin. Improvement of Stochastic Particle Swarm Optimization by Succession Strategy[J]. Communications of the Systemics and Informatics World Network, 2008, 3: 155-159.

[5] Cheng Jian, Yang Jie, Zhou Yue, et al. Flexible Background Mixture Models for Foreground Segmentation[J]. Image and Vision Computing, 2006, 24(5): 473-482.

编辑　顾逸斐