文章编号：0253-374X(2014)05-0790-08 DOI:10.3969/j.issn.0253-374x.2014.05.021

改进规则的可放缩矢量图形地图的查询模型

杜庆峰，卢冬蕊
(同济大学 软件学院，上海 201804)

摘要：分析了基于SVG(可放缩矢量图形)格式时态GIS(地理信息系统)领域，尤其是地理元素查询方面的研究现状，针对现有的XML(可扩展标记语言)查询方法进行了分析和对比.在基于SVG语法规则改进研究的基础上，并结合LISA Ⅱ(layered intersection scan algorithm Ⅱ，分层交叉扫描算法Ⅱ)查询技术，提出了一种适合改进规则的SVG格式的GIS数据地理元素查询模型及其算法，通过定义关键字查询语法规则表达式，定义关键字查询的“&”和“or”操作规则，实现了组合查询；通过定义实体元素节点，减少了节点数，提高了查询效率.该模型实现了SVG格式地图的地理元素查询，包括对整幅地图的各种组合查询和分图层的各种组合查询，解决了基于SVG格式地图元素查询的技术瓶颈.大量的数据验证表明，该查询模型是有效的.

关键词：地理信息系统；可放缩矢量图形语法规则改进；查询模型；算法
中图分类号：TP311.5 **文献标志码**：A

Query Model of Improved Rules Scalable Vector Graphics Map

DU Qingfeng, *LU Dongrui*
(College of Software Engineering, Tongji University, Shanghai 200092, China)

Abstract: Based on the scalable vector graphics (SVG) format, an analysis was made of the temporal geographic information system (GIS) field, especially current research status of geographic element query, and then a comparative study was made with the existing extensible markup language (XML) query methods. A SVG grammar rules-based research was proposed. In combination with the LISA Ⅱ query technology, a query model and the corresponding algorithm applicable for the GIS data geographic element of the improved-rules SVG format were proposed. A combination query was achieved by defining keyword query syntax regular expressions and the keyword query “&” and “or” operation rules. In addition, the number of nodes was reduced, and the query efficiency improved by defining the entity element node. Study results show that the model achieves geographic elements queries of SVG format map, including various combinations queries on the entire map and various combinations queries of layers. As a result, the technical bottleneck of query is solved based on SVG format map elements. Lots of data validate this query model is effective.

Key words: geographic information system (GIS); scalable vector graphics (SVG) grammar rules improvement; query model; algorithm

1 研究背景及问题的提出

1.1 SVG格式及相关研究工作

SVG(scalable vector graphics，可放缩的矢量图形)格式是W3C(World Wide Web Consortium，国际互联网标准组织)制定的一种新的2维矢量图形格式，严格遵从XML语法(extensible markup language，可扩展标记语言)，是一种和图像分辨率无关的矢量图形格式，因此，SVG格式图形已经被广泛地应用于GIS(geographic information system，地理信息系统)数据，这也使得基于SVG格式的GIS数据的查找问题成为了当下GIS领域研究的热点.因为SVG格式是严格遵循XML语法的，所以对于SVG格式的GIS数据的查找问题也就可以转化为对包含GIS数据的XML文档的查询问题.现有的XML文档的查询方法有两类：适合专业人员运用的XPATH(XML路径语言)和XQUERY(XML Query，XML查询)等XML相关的查询工具进行结构查询，以及适合普通用户的关键字查询方式.在

收稿日期：2013-07-08
基金项目：国家自然科学基金(41171303)
第一作者：杜庆峰(1968—)，男，教授，博士生导师，工学博士，主要研究方向为地理信息系统、软件工程、软件质量管理与软件测试. E-mail：du_cloud@sohu.com
通讯作者：卢冬蕊(1988—)，女，工程硕士，主要研究方向为地理信息系统. E-mail：zxhqhc@163.com

XML 关键字查询中，最关键的问题是查找包含所有关键字的最小 XML 片段，即 SLCA(smallest lowest common ancestor，最小最低公共祖先)问题[1]. 对于查找 XML 文档中 SLCA 的方法，现有的方法有许多，每种方法都有其优点和缺点. 最常见的算法有 STACK，ILE (indexed lookup eager)，SE (scan eager)，LISA(layered intersection scan algorithm，分层交叉扫描算法)，LISA Ⅱ(layered intersection scan algorithm Ⅱ，分层交叉扫描算法Ⅱ)等[1].

1.2　问题的提出

在对基于 SVG 格式的时态 GIS 的前期研究中，作者解决了 GIS 数据动态解析及增量存取机制的相关问题以及 SVG 格式地图的分层问题，包括：SVG 格式地图的解析，增量的存储机制，不同时间戳 SVG 格式地图差异匹配的算法，对 SVG 格式的语法规则进行增加分层信息的改进，DWG(DraWinG，AutoCAD 的一种图格式)格式地图到改进规则后的 SVG 格式地图的转换. 在已有研究成果的基础上，实现了地图的解析、存储、匹配和分层的相关功能，接下来对目前的瓶颈——改进规则后的 SVG 格式地图的查询技术进行研究.

2　查询模型的目标和思想

2.1　目标

本文的目标是在基于 SVG 语法规则改进研究的基础上，结合 LISA Ⅱ[1]查询技术和语义相关性的思想，建立一个适合改进规则的 SVG 格式 GIS 数据地理元素的查询模型. 通过定义关键字查询语法规则表达式、定义关键字查询的“&”和“or”操作规则实现组合查询，包括对整幅地图的各种组合查询和分图层的各种组合查询，并通过定义实体元素节点提高查询效率，从而解决基于 SVG 格式地图元素查询的技术瓶颈.

2.2　相关定义

2.2.1　SVG 语法规则改进技术

为了解决 SVG 格式 GIS 地图的分层问题，前期研究中对 SVG 格式的语法规则进行了改进，改进内容是增加分层信息的规则定义，为了模型研究的需要，这里加以描述.

遵循 XML 语法以及相关的 GIS 标准，对基于分层的 SVG 格式规则给出了如下定义：

```
/* * * * * * * * * * * * * * * * * * * * * * * * */
〈g id="_layer_Rivers"〉 SVG 层
/* SVG 元素定义
    * 〈g id="9999_00000000"〉 4×9→所属层的位置；8×0→子类中的标号
        * 〈SVG elements〉
        * 〈geo-attribute-data〉
            * 〈geox-NAME〉Value〈/geox-NAME〉
            * 〈geox-TYPE〉Value〈/geox-TYPE〉
        * 〈/geo-attribute-data〉
    * 〈/g〉
    * /
〈/g〉
/* * * * * * * * * * * * * * * * * * * * * * * * */
```

SVG 格式地图遵循 XML 语法，按照《基础地理信息要素分类与代码》的大类分层并编号；然后每一层再按照《基础地理信息要素分类与代码》的小类分层并编号，以此类推，直至不能再分层；最后，对每一个该层中 SVG 元素按照顺序编号，一个 SVG 元素包括基本信息以及属性数据，其中属性数据又包括属性名和属性类型. 图 1 即为改进规则后的包含分层规则的 SVG 文档. 图 2 为图 1 中 SVG 文档所对应的解析结构树. 其中，图 2 中的 bi 为 basic information 的缩写；v＝Test 表示 value＝Test；gad 为 geo-attribute-data 的缩写；gN 为 geox-NANE 的缩写；gT 为 geox-TYPE 的缩写.

2.2.2　查询模型相关数学规则的定义

定义 1　关键字语法表达式定义为

Q：[condition] “keyword”& [condition] “keyword” or [condition] “keyword”[2].

其中，[ ]中内容表示用户指定的相关语义条件，而“ ”中内容表示的是用户输入的关键字. 为了用户能查询到其最想要得到的查询结果，用户除了要输入想要查询的关键字以外，还要指定相关的语义条件，因此，在建立的查询模型中对用户输入的关键字查询语法表达式进行了定义[3].

此外，用户所输入的关键字是根据用户所要查询的内容而定的，并不一定精确地存在于 SVG 文档中. 因此，为了使用户能查找到其想要查询的内容，需要根据用户输入的关键字来匹配 SVG 文档中的相关内容，并给出相关提示，规则如下：① 按照《基础地理信息要素分类与代码》文件中的分类进行匹配；② 按照相似内容进行提示，如用户输入水，那么就要找到 SVG 中水系相关(河流、湖泊等)的内容提示给用户.

基于定义 1，有以下性质.

性质 1　$N(K_{i+1}) + \text{contain}(K_1, K_i) = \text{contain}(K_1, K_i, K_{i+1})$[4].

```
〈? xmlversion="1.0"encoding="utf-8"?〉
〈! DOCTYPEsvg PUBLIC "-//W3C//DTD SVG 1.1//EN""http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd"〉
〈svgxmlns="http://www.w3.org/2000/svg"xmlns:xlink="http://www.w3.org/1999/xlink"viewBox="0 0 750 750"〉
〈gtransform="matrix(1,0,0,-1,-50,800)"〉
〈gid="_layer_Rivers"〉
    〈gid="0002_00000001"〉
        〈text x="598.8" y="703.8" fill="  0000FF" font-size="37.1" transform="matrix(1,0,0,-1,0,1370.5) "〉Test〈/
text〉
        〈geo-attribute-data /〉
    〈/g〉
〈/g〉
〈g id="_layer_Roads"〉
    〈g id="0003_00000001"〉
        〈polyline points="50,50 800,50 800,800 50,800 50,50" stroke="  FF0000" fill="none"〉
        〈/polyline〉
        〈geo-attribute-data〉
            〈geox-NAME〉Test〈/geox-NAME〉
            〈geox-TYPE〉Super〈/geox-TYPE〉
        〈/geo-attribute-data〉
    〈/g〉
〈/g〉
...
〈/g〉〈/svg〉
```

图1 改进规则后的SVG文档

Fig.1 Improved-rules SVG document

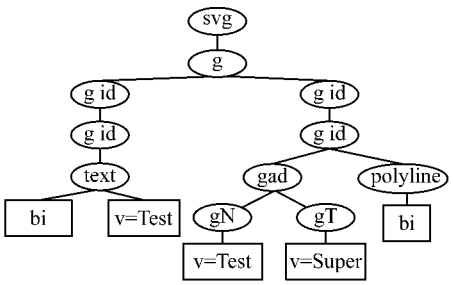

图2 改进规则后的SVG解析树

Fig.2 Improved-rules SVG parsed tree

按照用户输入的关键字语法表达式，从关键字 K_1 开始，将找到的包含关键字 K_i 的节点加入到contain()中，若表达式中的关键字 K_{i+1} 与 K_i 由“&”符号相连，则将 K_{i+1} 所在节点继续添加到contain()中，记为：$\mathrm{N}(K_{i+1})+\mathrm{contain}(K_1,K_i)=\mathrm{contain}(K_1,K_i,K_{i+1})$. 若表达式中的关键字 K_{i+1} 与 K_i 由“or”符号相连，则将 K_{i+1} 所在节点重新加入到一个新的contain()中.

定义2 关键字语义条件：ksc(keyword semantic condition)，记为 ksc(k)[5].

关键字语义条件反应了关键字节点之间的语义相关性[6]. 例如，对于Q：[name]“Test”&[type]“Super”有两个关键字“Test”和“Super”，那么对于“Test”和“Super”的关键字语义条件为：ksc(Test)=[name]，ksc(Super)=[type].

在查询模型中，通过实体元素节点来找到符合ksc的关键字节点[6-8]，实体元素节点的定义如下.

定义3 实体元素节点.

在前期的研究中，将改进规则后的SVG格式所对应的XML结构树节点分为元素节点、文本节点和属性节点[8]. 在此基础上，把SVG格式GIS数据对应的结构树中包含属性、包含文本及既包含属性又包含文本的元素节点及其属性或文本看成一个整体，称之为一个实体元素节点[6-8].

通过实体元素节点找到符合ksc的关键字节点[6-8]的方法为：查看包含关键字 k 的实体元素节点中是否有满足ksc(k)的相关内容，如果有，那么称该节点为符合条件语义的节点.

定义4 改进的LISA Ⅱ方法数学模型为：

$$\bigcup_{i}^{\mathrm{id}}\left\{\bigcup_{j}^{\min\max L(D_1,D_2,\cdots,D_k)\to 3}(D)\left[\bigcap(\dot{S}_i(j)\right]\right\}$$

$i\in$ id(图层)，$j\in$ level(结构树层)

其中：$\cup$ 表示并集操作；$\cap$ 表示交集操作；D 表示delete已找到的层的SLCA(smallest lowest common ancestor，最小最低公共祖先)的操作；id表

示图层；level 表示结构树的层. min max $L(D_1, D_2, \cdots, D_k)$表示结构树中需要做交集的节点所在的最低层(由于已经转换成实体元素节点，所以按照文献[1]，已经将 min max $L(D_1, D_2, \cdots, D_k)-1$ 变成了 min max $L(D_1, D_2, \cdots, D_k)$)；$\dot{S}_i(j)$表示 S 中第 i 图层的节点编号所在的第 j 层的相应节点. 具体如下：

(1) 按照文献[1]中的 LISA Ⅱ 方法，将包含关键字并符合语义条件的实体元素节点编码，假设将包含关键字“aaa”的节点和包含关键字“bbb”的节点进行编码，如图 3 所示，只编码到第 min max $L(D_1, D_2, \cdots, D_k)$层，也就是图中的第 5 层.

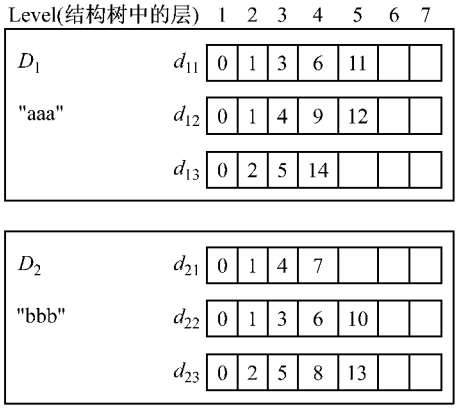

图 3　包含关键字并符合语义条件的实体元素节点编码

Fig. 3　Entity element nodes coding including keywords and meeting the semantic conditions

(2) 将这些节点按照第 2 层(见改进规则后的 SVG 格式对图层类别的定义)的编号(或者是节点的 id)，即所在的图层进行分类，并将编号缩短至第 3 层，例如：图 4 中 01→roads 层和图 5 中 02→buildings 层.

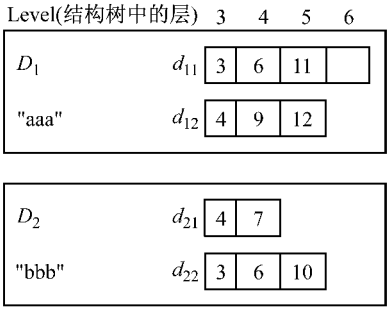

图 4　分类后 roads 类的节点编码

Fig. 4　Node coding of class roads after classification

(3) 针对每个类别，从第 5 层开始做交集运算，以图 5 为例，即数学模型中的$\bigcap(\dot{S}_i(5))$操作，其中 $i \in$roads. 如果结果为空集，则往前一层进行交集运算，如图5，$\bigcap(\dot{S}_i(4))=\{6\}$，交集不为空，那么6即

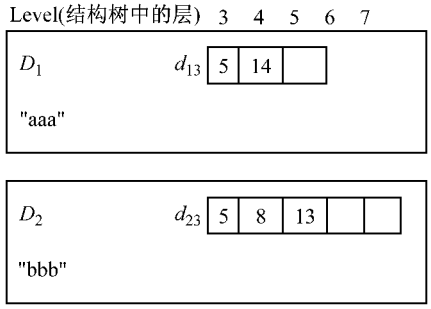

图 5　分类后 buildings 类的节点编码

Fig. 5　Node coding of class buildings after classification

为一个 SLCA. 接着继续往前一层，在往前一层进行交集运算前，先要删除掉之前所找到的包含 SLCA “6”的节点 d_{11} 和 d_{22}，即数学模型中的(D)操作. 往前第 3 层只剩下 d_{12} 和 d_{21} 两个节点，对应的$\bigcap(\dot{S}_i(3))=\{4\}$，交集不为空，又得到一个 SLCA“4”，发现此时已经到了最小的第 3 层. 因此，停止交集运算. 然后，将得到的各层的 SLCA 作并集找到所有 roads 的图层中所有符合条件且包含关键字的最小 XML 片段，即数学模型中的 $\bigcup_{j}^{\min\max L(D_1, D_2, \cdots, D_k)\to 3}(D)[\bigcap(\dot{S}_i(j))]$.

(4) 对 buildings 层做同样的运算并找到该图层中所有符合条件且包含关键字的最小 XML 片段，然后将各个图层的结果进行并集运算，从而找到 SVG 格式 GIS 数据文档中所要查询的结果，即数学模型中的 $\bigcup_{i}^{\mathrm{id}}\{\cdots\}$运算.

2.3　查询模型思想

基于对 SVG 格式所改进的分层信息的规则定义以及前文相关的数学规则定义，查询模型的主要思想为：根据用户输入的关键字查询语法表达式，找出解析结构树中相应的关键字节点，然后转化为实体元素节点，筛选出符合相关条件的节点，按图层分类，针对每一图层，按照 LISA Ⅱ[1] 方法，从结构树的最深层开始向上递归，通过交集运算找到相应的 SLCA 节点，最后将得到的 SLCA 节点进行并集运算，从而得到最终的查询结果.

基于查询模型的主要思想，具体的逻辑步骤为：① 根据用户输入的关键字查询语法表达式，对于每个关键字 K_i，找到 SVG 解析结构树中所有包含查询关键字 K_i 的节点集合 R_i；② 将 SVG 格式 GIS 数据对应的解析结构树转化为实体元素节点结构树，并针对每个关键字 K_i，找到相应的包含关键字 K_i 的实体元素节点集合 R_i^*；③ 对于每个关键字 K_i，

在相应的R_i^*集合中找到满足关键字查询语法表达式中ksc(K_i)条件的节点集合S_i;④ 根据关键字查询语法表达式中"&"和"or"的语法规则,将每个S_i集合中的节点按照定义1中的性质分别加入到不同的contain( )集合中并将得到的每个contain( )集合记为S;⑤ 对于每个S中的节点按照改进规则后的SVG格式图层的概念进行分类,其中对于每一图层类中的S节点,根据定义4所描述的数学模型,从相应的SVG解析结构树的最深层开始向上递归,通过交集运算找到相应的SLCA节点,并将找到的每个图层类的SLCA节点进行并集运算;⑥ 最后将每个S集合所得到的SLCA节点集合进行并集,从而得到最后的结果集Result.

以上步骤所构建的查询模型流程图如图6所示.

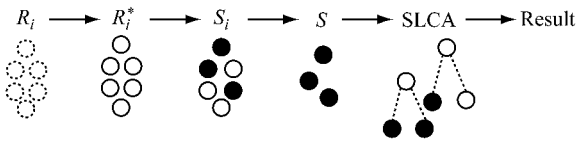

图6 基于改进规则的SVG格式GIS数据的查询模型流程图

Fig.6 Query model flow diagram based on improved-rules SVG format GIS data

3 查询模型算法实现

3.1 算法的实现

针对上述基于改进规则的SVG格式GIS数据的查询模型,将其转化成相应的查询算法,具体算法步骤如下:

步骤1 分析用户输入的关键字语法表达式,根据每个关键字,匹配到相应的SVG文档所对应的解析结构树中相关的内容提示给用户,由用户决定是否符合其查询意图并确定返回给算法.根据用户的反馈,通过相关的内容找到结构树对应的节点,然后转化为相应的实体元素节点并进行编码.

步骤2 按照用户指定的相关条件,筛选出符合语义条件的相关节点.

步骤3 分析用户输入的表达式,将由"&"符号相连的关键字归结在一起,由"or"符号相连的关键字分开进行查找.

步骤4 根据解析结构树中节点的id号或者节点编码中第2层的编码号找到节点所对应的图层进行分类.

步骤5 将同一类中包含不同关键字的节点集合从SVG解析结构树的最深层min max $L(D_1,D_2,\cdots,D_k)$开始,获取集合中对应该层的所有编码进行交集运算,如果交集不为空,则其中对应的节点就是一个SLCA节点.在进一步求解相应的解析结构树较低层的SLCA节点时,将前一层所得的SLCA节点所对应的原关键字节点从集合中排除出去,然后在更低层中重复与最深层一样的交集计算,直到同一关键字的集合为空或者到达第3层为止.

步骤6 将解析结构树每一层中找到的SLCA节点合并.

步骤7 将解析结构树每一类图层中找到的SLCA节点合并.

步骤8 将由"or"符号连接的关键字所查找到的SLCA节点合并,得到最终的节点集结果.

最后根据得到的节点集结果就可以得到想要查询的XML片段,即最终的查询结果.

逻辑实现——基于改进规则的SVG格式地图查询算法(改进的LISA Ⅱ算法)如图7所示.

3.2 算法复杂度分析

3.2.1 时间复杂度计算

改进规则的SVG格式GIS地理元素查询算法中,设SVG中节点个数为n,实体元素节点个数为m,带有关键字K_1到K_t的实体元素节点的个数分别为m_1到m_t,其中符合语义条件的个数为c_{m_1}到c_{m_t}.

有序整数集合的交集运算算法如文献[1],设有最小个数的关键字实体元素节点集合所包含的符合语义条件的节点个数为c_m,那么总的时间复杂度为

$$O\Big[\sum_{i=2}^{t}(\max\{c_m,|\mathrm{med}_{1i}|\})\Big]$$

在求解SLCA的改进LISA Ⅱ算法中,算法第1行到第2行,找到包含关键字的节点并转化为实体元素节点,要对SVG中的节点进行遍历,时间复杂度为$O(n)$.算法第3行,找到符合条件的实体元素节点需要对包含关键字的实体元素节点进行遍历,时间复杂度为$O(m_1+\cdots+m_t)$.算法第4行到第12行,将S_1到S_t中的节点添加到S中,时间复杂度为$O(c_{m_1})+\cdots+O(c_{m_t})$.算法第13行到第29行,参照文献[1]以及有序整数集合的交集运算的时间复杂度[1],总的时间复杂度为

$$O\Big[(\min\max L-2)\Big\{\sum_{i=1}^{t}(c_{m_i}+c_{m_i}\log c_{m_i})+$$

```
1.  find set R of nodes with all the keywords(K).
2.  transform set R into set R* and transform all nodes into integer codes. //转换成实体元素节点并编码
3.  select nodes which match ksc(k_i) and store them in set S_i. //筛选出满足条件的节点
4.  for each K{
5.  if(K_i+1&K_i) //将由"&"相连的关键字节点放到同一个集合中
6.  S_i+1+contain(K_1,K_i)=contain (K_1,K_i,K_i+1)
7.  set contain(K_1,K_i,K_i+1) as S
8.  else if(K_i+1 or K_i) //将由"or"相连的关键字节点放到不同集合中
9.  new contain();
10. S_i+1+contain()=contain(K_1)
11. set contain(K_1) as S
12. }
13. for each S{
14. for each id, store them according to keywords, marked as xIS_i//按照图层分类
15. compute max L_1(IS_i) for each IS_i//计算节点深度
16. min max L=min{ max L_i|1≤i≤k} //表示结构树中需要做交集的节点所在的最低的层
17. v={};tmp={}
18. for j=min max L to 3
19. while no IS_i is null do { //查找 SLCA 节点
20. build integer set S'_i(1<i<k) corresponding level j and sort it, S'_i is the smallest set
21. tmp = S'_i
22. for i=2 to k
23. tmp=intersection(v, tmp) //编码做交集运算(子算法:有序整数集合的交集运算算法[1])
24. v +=tmp;
25. delete nodes which have integers contained in v, from each IS_i//删除已找到的 SLCA 节点所对应的原关键字节点
26. }
27. return SLCA nodes corresponding to v //返回找到的 SLCA 节点
28. merge SLCA nodes of each id //合并各图层的 SLCA 节点
29. }
30. merge SLCA nodes of each S //合并由"or"相连的关键字所找到的相应的 SLCA 节点
```

图 7　基于改进规则的 SVG 格式地图查询算法

Fig.7　Query algorithm based on improved-rules SVG format map

$$\sum_{i=2}^{t}(\max\{c_m,|\mathrm{med}_{1i}|\})\Big\}\Big]$$

因为算法第 28 行到第 30 行的运算代价较小，所以在复杂度的计算公式中忽略不计，因此基于以上分析，改进规则的 SVG 格式 GIS 地理元素查询算法总的时间复杂度为

$$O(m_1+\cdots+m_t)+O(c_{m_1})+\cdots+O(c_{m_t})+O\Big[(\min\max L-2)\Big\{\sum_{i=1}^{\tau}(c_{m_i}+c_{m_i}\log c_{m_i})+\sum_{i=2}^{\tau}(\max\{c_m,|\mathrm{med}_{1i}|\})\Big\}\Big]\approx O\Big[(\min\max L-2)\Big\{\sum_{i=1}^{t}(c_{m_i}+c_{m_i}\log c_{m_i})+\sum_{i=2}^{t}(\max\{c_m,|\mathrm{med}_{1i}|\})\Big\}\Big]$$

3.2.2　空间复杂度计算

S 所需的空间复杂度为 $O(S_1+\cdots+S_i)=O(c_{m_1}+\cdots+c_{m_t})=O(\sum c_m)$，因为最终得到的 SLCA 节点不会超过 $O(\sum c_m)/2$，所以所占空间很小. 因此，最终改进规则的 SVG 格式 GIS 地理元素查询算法的空间复杂度约为 $O(\sum c_m)$.

4　查询算法的验证

针对改进规则的 SVG 格式地图的查询模型及算法，下面以一个改进规则后的 SVG 数据文件查询为例对其进行验证. 验证过程包括被查询的改进规则的 SVG 文件解析后的结构树，用户输入的关键字语法表达式和算法的整个执行过程.

4.1　SVG 文件对应的解析树

图 8 为用作验证的 SVG 文件所对应的解析结构树，其中 g 节点的 id 值与 line，circle 和 polyline 的值省略.

4.2　用户输入的关键字查询表达式

[name]"Heping"& [type]"Super" or [name]"Xingfu"& [type]"Normal".

4.3　算法执行过程

第1步，找到包含关键字"Heping"，"Super"，

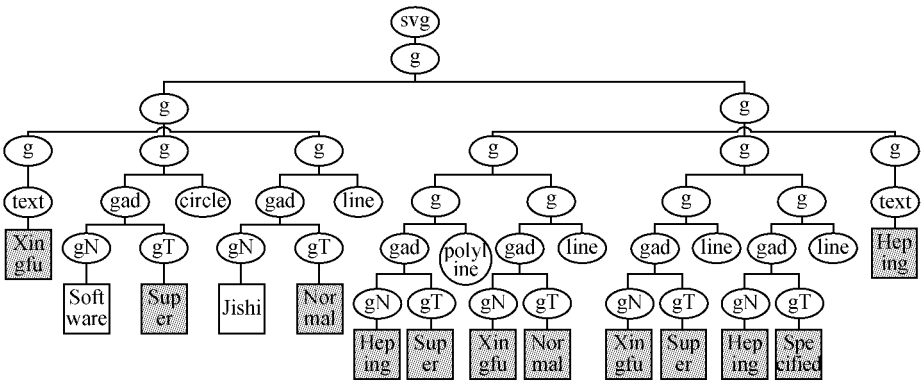

图 8 SVG 对应的解析结构树

Fig.8 The parsed structural tree corresponding with SVG

"Xingfu"和"Normal"的节点,即图 8 中灰色方框所示的节点.

第 2 步,转化为实体元素节点并按照前序遍历对节点进行编号,如图 9 所示.

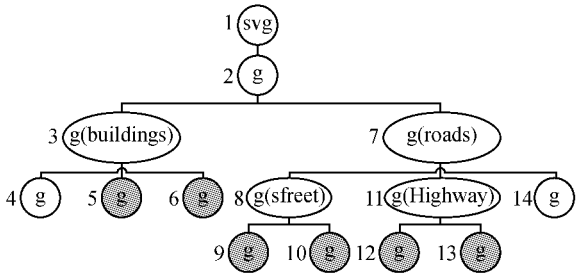

图 9 转化为实体元素节点并进行编号的 SVG 解析结构树

Fig.9 SVG parsed structural tree after changing to entity element nodes and coding

第 3 步,找到符合条件 ksc(Heping)=name, ksc(Super)=type, ksc(Xingfu)=name 和 ksc(normal)=type的节点,即图 9 中灰色方框所示的节点.

第 4 步,按照用户输入的查询表达式,[name]"Heping"和[type]"Super"以及[name]"Xingfu"和[type]"Normal"是由"&"相连的,而两组之间是由"or"相连的,所以把由"&"相连的节点放到同一个集合中,即满足[name]" Heping"和[type]" Super"的节点放到集合 1 中,集合 1 包括: Heping——9, 13; Super——5,9,12. 同理,满足[name]"Xingfu"和[type]"Normal"的节点放到集合 2 中,集合 2 包括: Xingfu——10,12;Normal——6,12.

第 5 步,对每个集合中的节点按照所属的图层分类. 集合 1 包括: buildings(Heping——null; Super——5)和 roads(Heping——9,13; Super——9,12). 集合 2 包括: buildings(Xingfu——null; Normal——6)和 roads(Xingfu——10, 12; Normal——12).

第 6 步,对集合 1、集合 2 中的节点进行 LISA Ⅱ处理. 由于集合 1、集合 2 中 buildings 层的关键字节点并非都存在,所以可以忽略 buildings 层,这样,只要在 roads 层查找 SLCA 节点即可. 根据集合中节点所在深度,计算得两组集合的 min max $L(D_1, D_2)=5$. 这样两组集合只要从第 5 层开始进行交集运算直到第 3 层为止即可,如图 10 和图 11 所示. 那么集合 1 从第 5 层开始,$D_1=\{9,13\}$,$D_2=\{9,12\}$,两个整数集合的交集为{9},因此,节点 9 为一个 SLCA 节点. 然后删除节点 9 对应的关键字节点,之后进入第 4 层,此时,$D_1=\{11\}$,$D_2=\{11\}$,交集为{11},因此,节点 11 也是一个 SLCA 节点. 接着,删除节点 11 对应的关键字节点. 而此时,集合 1 中的 D_1,D_2 都已经为空了,那么 LISA Ⅱ算法终止. 最后得到的结果为{9,11}. 同理,集合 2 从第 5 层开始,$D_1=\{10,12\}$,$D_2=\{12\}$,两个整数集合的交集为{12},因此,节点 12 为一个 SLCA 节点. 然后删除节点12对应的关键字节点,之后进入第4层,此时,

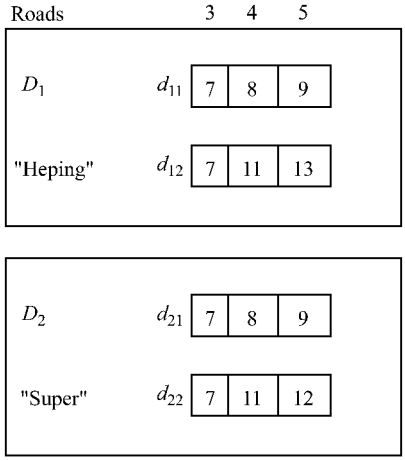

图 10 集合 1 节点编码

Fig.10 Nodes coding of set 1

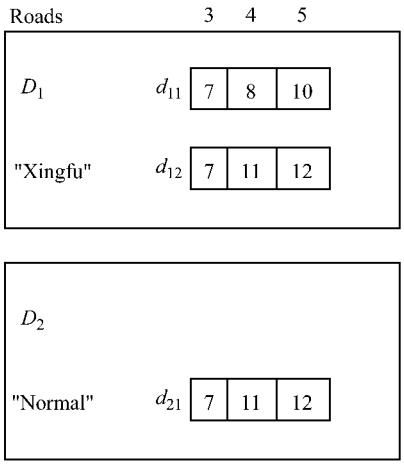

图 11　集合 2 节点编码

Fig.11　Nodes coding of set 2

$D_1=\{8\}$，$D_2=\{\ \}$，由于 D_2 已经为空，那么 LISA II 算法终止. 最后得到的结果为{12}. 最后，合并集合 1 与集合 2，得到的最终结果{9，11，12}即为最后的 SLCA 节点结果集.

5　结论

论文在改进规则的 SVG 格式地图的基础上，对 GIS 数据地理元素的查询提出了相关的模型及其算法. 该模型及算法主要在以下几个方面有所创新：①应用改进规则后的 SVG 格式，使得解析树结构清晰，层次分明，十分有利于 GIS 元素的查找，提高了效率；②定义了用户输入关键字查询语法规则表达式和关键字查询的“&”和“or”操作规则，实现了组合查询；③定义了实体元素节点，减少了节点数，提高了查询效率；④在 LISA Ⅱ方法的基础上，使用了按照图层进行查找的方法，减少了交集的次数以及排序的时间；⑤实现了 SVG 格式地图的地理元素查询，包括对整幅地图的各种组合查询和分图层的各种组合查询，解决了基于 SVG 格式地图元素查询的技术瓶颈；⑥算法复杂度低，适合对大规模 SVG 格式地图进行地理元素的查找.

此外，该算法在查询效率、准确度和空间占用量方面有如下结论：①由于将 SVG 格式进行了分层的定义，并鉴于时间复杂度分析，所以对于海量地图数据，查询效率是较高的. 查询数据由大量的 DWG 格式地图转换而来，所以数据准备也并不耗时. ②由于算法将 SVG 格式进行了分层的定义，没有涉及到模糊查询，只是用模拟的地图数据来进行查询验证，所以得到的查询结果准确率较高. ③算法的内存空间占用量取决于所要查询的 SVG 地图数据的大小.

该查询模型与算法为今后基于 SVG 的 GIS 地理元素的模糊查找等相关研究提供了新的思路，奠定了基础. 最后通过大量的数据验证表明该算法是有效的.

参考文献：

[1] 孔令波，唐世渭，杨冬青，等. XML 信息检索中最小子树根节点问题的分层算法[J]. 软件学报，2007，18(4)：919.
KONG Lingbo，TANG Shiwei，YANG Dongqing，et al. Layered solution for SLCA problem in XML information retrieval[J]. Journal of Software，2007，18(4)：919.

[2] 孔令磊. 面向 XML 文档的关键字查询的研究[D]. 北京：北京交通大学，2008.
KONG Linglei. Research on XML document oriented keyword query[D]. Beijing：Beijing Jiaotong University，2008.

[3] 黎军. 综合文档语义与用户查询语义的 XML 关键字查询研究[D]. 重庆：西南大学，2011.
LI Jun. Research on XML keyword query of comprehensive document semantic and user query semantic[D]. Chongqing：Southwest University，2011.

[4] 公爱国，石冰，周钦亮. 一种基于关键字查询的 XML 检索模型[J]. 计算机科学，2006，33(10)：67.
GONG Aiguo，SHI Bing，ZHOU Qinliang. An XML search model based on keyword[J]. Computer Science，2006，33(10)：67.

[5] 李辛. 基于语义相关性的 XML 关键字查询的研究与实现[D]. 北京：北京交通大学，2009.
LI Xin. Research and implementation of XML keyword query based on semantic relevance[D]. Bejing：Bejing Jiaotong University，2009.

[6] 曾晓宁，蔺旭东，李密生，等. 一种基于节点语义相关性的 XML 关键字查询算法[J]. 电脑知识与技术，2009，5(11)：2888.
ZENG Xiaoning，LIN Xudong，LI Misheng，et al. An XML keyword query algorithm based on node semantic relevance[J]. Computer Knowledge and Technology，2009，5(11)：2888.

[7] 姚全珠，余训滨. 基于最小相关实体子树的 XML 关键字查询算法[J]. 计算机应用，2012，32(4)：1090.
YAO Quanzhu，YU Xunbin. XML keyword search algorithm based on smallest lowest entity sub-tree interrelated[J]. Journal of Computer Applications，2012，32(4)：1090.

[8] Du Q，Guo Z C，Tang X. DiffSvg-matching algorithm of different timestamp maps based on SVG[C]//IEEE International Conference on Computer Science and Service System. [S. L.]：IEEE，2012：167-175.