一种可扩展的 XPath 查询最小化算法框架*)

林 峰[1] 冯建华[1] 塔 娜[1] 李国良[1] 洪 亲[2]

(清华大学计算机科学与技术系 北京 100084)[1] (福建师范大学物理与光电信息科技学院 福州 350007)[2]

摘 要 XPath是XML的基本查询语言,XPath查询最小化对于提高XML数据库的查询性能具有重要意义。但是,由于XPath查询最小化是一个coNP完备问题,大部分已有的算法局限于处理简单的XPath片段。本文从一个新的角度入手,综合考虑完备性和高效性,提出了一个新的查询最小化框架,与已有算法"面向结点",即逐个删除冗余结点的解决思路不同,本文提出"面向树模式"的方式,即通过计算树模式的自同态映射,寻找目标结点集最小的自同态映射,进而求解最小等价查询树的方法。该方法具有较高的效率,而且在一定情况下是完备的,尤其是可以进一步扩展到更复杂的XPath片段。本文以此框架为基础,给出一个可以计算复杂查询模式的算法。

关键词 XPath,查询,自同态,最小化

An Extensible Framework for XPath Query Minimization

LIN Feng[1] FENG Jian-Hua[1] TA Na[1] LI Guo-Liang[1] HONG Qin[2]

(Department of Computer Science and Technlogy, Tsinghua University, Beijing 100084)[1]

(School of Physics and OptoElectronics Technology, Fujian Normal University, Fuzhou 350007)[2]

Abstract XPath is a standard language for XML queries, and XPath query minimization is very important for improving the performance of XML queries. However, XPath query minimization is a coNP-complete problem, and most previous algorithms are restricted to simple XPath fragments. We study the problem from a novel perspective, and present an efficient and extensible framework of query minimization. Traditional methods, which are "node oriented", always delete redundant node one by one, however we present an alternative way of "tree pattern oriented". We first compute endomorphism of the initial tree pattern, then find the endomorphism with the smallest target nodes collection size, and finally retrieve the minimal equivalent query pattern tree. Our method is efficient, and is complete under certain conditions. More importantly, it can be extended to more complex XPath fragments. We also give an algorithm, which is adaptive to complex query patterns based on the new framework.

Keywords XPath, Query, Endomorphism, Minimization

1 引言

XML[1]已经成为网络上数据表示、传输、交换的标准。XPath[2]是XML的基本查询语言,也是更复杂的查询语言XQuery[3]的主要组成部分。XPath查询最小化对于加快XML数据库的查询具有重要意义。

XPath查询经常使用树模式(Tree Pattern)来表示,XPath查询的计算过程就是树模式与XML文档的匹配过程,因此树模式的规模直接与查询效率相关,较小的树模式意味着较高的查询效率。[4]但是,由于下述原因,树模式通常包含冗余结点。首先,构造最小化查询需要很强的技巧,普通用户难以给出兼顾正确性和高效性的查询语句。其次,对于广泛存在的分布式查询系统,一般的查询分解方法会引入冗余结点。

简单地说,在保持树模式等价性的前提下,删除其中的冗余结点,称为树模式最小化,后文会给出更加准确的定义。常见的XPath片段包括:孩子轴(/)、后裔轴(//)、谓词([])和通配符(*),包含上述4种运算的XPath片段记为$XP^{\{/,//,[],*\}}$。类似地,仅包括孩子轴和后裔轴的XPath片段记为$XP^{\{/,//\}}$。

简单的XPath片段,例如$XPath^{\{/,[],*\}}$的最小化问题可以在多项式时间内解决[5],另一个XPath片段$XP^{\{/,//,*\}}$也可以在多项式时间内得到最小等价查询[6],对于没有"*"的XPath片段$XP^{\{/,//,[]\}}$,同样可以在多项式时间内最小化[7]。然而,与上述结论完全不同的是,如果一个XPath片段包含了前面提到的4种运算符,即$XP^{\{/,//,[],*\}}$,那么,其最小化问题的复杂度为coNP完备[5]。文[8]列出了各种XPath片段最小化的时间复杂度。这意味着计算最小等价查询的代价是巨大的。

因为查询最小化的目的是提高查询性能,所以查询最小化本身的算法复杂度不能太高,否则将失去优化的意义。查询最小化算法应该在尽量兼顾完备性的基础上,更加重视执行效率。本文旨在提出一种注重实际效率的查询最小化问题的解决方案,主要贡献如下:(1)提出了一个以同态映射为基础的新的查询最小化的框架。该框架对于$XP^{\{/,[],*\}}$、$XP^{\{/,[],//\}}$片段,可以得到最小等价查询语句;对于$XP^{\{/,//,[],*\}}$

*)本论文得到福建省科技计划项目的资助,资助号:2006F501050079。林 峰 硕士研究生,主要研究方向为XML数据库;冯建华 副教授,主要研究方向为数据库、XML数据库和WWW环境下的信息处理;塔 娜 硕士研究生,主要研究方向为XML数据库缓存技术;李国良 博士研究生,主要研究方向为复杂数据上的关键字搜索、XML查询处理、数据挖掘、数据集成、XML数据库;洪 亲 高级工程师,主要研究方向为数据库、XML数据库。

片段，具有较高的效率，结果接近最小化查询语句。(2)提出一个具体的最小化算法。对于 $XP^{\{/,[],//\}}$ 片段，该算法的复杂度为 $O(n^2)$。(3)上述算法具有可扩展性。与以往的算法不同，对于 $XP^{\{/,//,[],*\}}$ 片段，仍然能够保持其高效性。

本文其它部分组织如下：第 2 节是基本概念和定义，第 3 节给出了 XPath 查询最小化算法框架的理论基础，第 4 节是算法简介，最后是结论和展望。

2 相关定义

2.1 XML 文档树

定义 1(XML 文档树) 我们通常用树来表示 XML 文档，称之为 XML 文档树[5]。

XML 文档树结点标签(Label)来自于无限字符集 Σ，Σ 中的字符表示元素标签、属性标签，以及 XML 文档中出现的文本。一棵 XML 文档树 t 的结点集记为 $NODES(t)$，边集记为 $EDGES(t)$，根结点记为 $ROOT(t)$。对于 t 的结点 x，其标签记为 $LABEL(x)$，$LABEL(x)\in\Sigma$。$EDGES(t)$ 的传递闭包记为 $EDGES^+(t)$，

$$EDGES^+(t)=EDGES(t)\cup EDGES(t)\cdot EDGES^+(t)$$

其中“·”表示连接运算。结点 x、y 具有父子关系，记为 $(x,y)\in EDGES(t)$；结点 x、y 具有祖先-后裔关系，记为 $(x,y)\in EDGES^+(t)$。文档树 t 的规模用 t 的结点数目表示，记为 $|t|$。

2.2 树模式

本文研究的 XPath 片段查询语句 q 的语法如下：

$$q\rightarrow n\mid *\mid .\mid q/q\mid q//q\mid q[q]$$

其中 $n\in\Sigma$，“*”表示通配符，“.”表示当前结点，“/”表示孩子轴，“//”表示后裔轴，“[]”表示谓词。

定义 2(树模式) 我们用树模式(Tree Pattern)来表示查询语句。一个维度为 $k(k\geqslant 0)$ 的树模式 p 是一棵树，其结点标签集为 $\Sigma\cup\{*\}$，p 的边分为两种，一种称为后裔边(Descendant edge)，另一种称为孩子边(Child edge)，p 的输出结点(Output nodes)的数目为 k。

$NODES(p)$、$EDGES(p)$、$ROOT(p)$、$LABEL(x)$($x\in NODES(p)$)、$|p|$ 的含义与 2.1 节所述相似。后裔边集记作 $D\text{-}DEGES(p)$，孩子边集记作 $C\text{-}EDGES(p)$。$EDGES(p)=D\text{-}DEGES(p)\cup C\text{-}EDGES(p)$。在树模式的图形表示中，用双线表示后裔边，用单线表示孩子边。

每个 XPath 表达式都可以转换为树模式。因为 XPath 查询语句只有 1 个输出结点，所以由 XPath 查询转换的树模式的维度为 1[9]。

定义 3(嵌入) 给定树模式 p 和文档树 t，定义从 p 到 t 的嵌入(Embedding)e：$NODES(p)\rightarrow NODES(t)$，满足如下 4 个条件：

1) $e(ROOT(p))=ROOT(t)$；

2) $\forall x\in NODES(p)$，$LABEL(x)=*$ 或 $LABEL(x)=LABEL(e(x))$；

3) $\forall (x,y)\in C\text{-}EDGES(p)$，$(e(x),e(y))\in EDGES(t)$；

4) $\forall (x,y)\in D\text{-}EDGES(p)$，$(e(x),e(y))\in EDGES^+(t)$。

给定维度为 k 的树模式 p 和文档树 t，$X=(x_1,x_2,\cdots,x_k)$ 是 p 的 k 元输出结点，定义 $p(t)$ 为 p 在 t 上的查询结果：

$$p(t)=\{e(X)\mid e\text{ 是从 }p\text{ 到 }t\text{ 的嵌入}\}$$

文档树 t、树模式 p 和嵌入 e 的一组实例如图 1 所示，其中带圈的结点是输出结点，虚线表示嵌入关系。

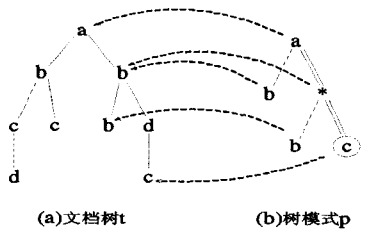

图 1 文档树、树模式、嵌入举例

2.3 布尔模式

树模式包含了两类信息：树结构和返回结点。为了简化问题，我们考虑布尔模式，其定义如下：

定义 4(布尔模式) 无返回结点的树模式，称为布尔模式(Boolean pattern)。

如果 p 是布尔模式，那么 $p(t)=\varnothing$ 或 $p(t)=\{()\}$。对于第一种情况，我们认为 $p(t)$ 返回假；对于第二种情况，我们认为 $p(t)$ 返回真。实际上，布尔模式与普通树模式的包含问题是等价的[9]，因此本文其余部分只讨论布尔模式。

2.4 树模式的包含、等价和最小化

为了定义树模式最小化，我们首先给出树模式包含和等价的定义：

定义 5(树模式包含) 给定两个树模式 p_1 和 p_2，p_2 包含 p_1(或 p_1 被 p_2 包含)，当且仅当对任意文档树 t，满足 $p_1(t)\subseteq p_2(t)$，记为 $p_1\subseteq p_2$。

定义 6(树模式等价) 两个树模式 p_1 和 p_2 是等价的，当且仅当 $p_1\subseteq p_2$ 且 $p_2\subseteq p_1$，记为 $p_1\equiv p_2$。

定义 7(树模式最小化) 给定树模式 p，记 p 的最小化树模式为 p_{min}，则 p_{min} 满足 $p_{min}\equiv p$，且不存在 p' 满足 $p'\equiv p$ 使 $|p'|<|p_{min}|$。计算 p_{min} 的过程称为树模式最小化。

根据上述定义，最小化问题实质是等价问题，也就是包含问题。

2.5 同态映射

为了求解树模式最小化问题，我们使用一种包含映设工具——同态映射，它的定义如下：

定义 8(同态映射) 从树模式 q_2 到 q_1 的同态映射(Homomorphism)，也称同态，$h:q_2\rightarrow q_1$，把 q_2 的结点映射到 q_1 的结点，同时满足如下 4 个条件：

1) $h(ROOT(q_2))=ROOT(q_1)$；

2) $\forall x\in NODES(q_2)$，$LABEL(x)=*$ 或 $LABEL(x)=LABEL(h(x))$；

3) 如果 $(x,y)\in C\text{-}EDGES(q_2)$，那么 $(h(x),h(y))\in C\text{-}EDGES(q_1)$；

4) 如果 $(x,y)\in D\text{-}EDGES(q_1)$，那么 $(h(x),h(y))\in EDGES^+(q_1)$。

同态映射的一个简单实例如图 2 所示，虚线表示同态映射关系。

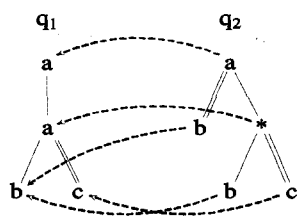

图 2 从树模式 q_2 到树模式 q_1 的同态映射

在关系数据库中，存在同态映射是合取查询(Conjunctive query)包含的充要条件[10]。对于简单的树模式，如 $XP^{\{/,//,[]\}}$ 和 $XP^{\{/,[],*\}}$，同态映射仍然是查询包含的充要条件[7]，因此同态映射可以检验这类查询的等价性。但是，对于一般的树模式，同态映射仅仅是充分条件，而非必要条件[9]。

目前已有的根据同态映射理论设计的算法，如文[7,11]，只能应用于 $XP^{\{/,//,[]\}}$ 片段，不能扩展到其他片段上。我们提出的查询最小化框架具有可扩展性，能够方便地应用于其他XPath片段。

3 理论基础

在这一部分，我们将提出一个新的树模式最小化框架。已有的算法是"面向结点"的[7,11]，即遍历原树模式各个结点，如果该结点是冗余的，那么从树模式中删除，直到消除所有的冗余结点，获得最小等价树模式。我们从另一个角度考虑最小化问题，即"面向树模式"的方式，相关定义和理论如下。

定义 9(自同态) 从树模式 q 到其自身的同态，称为自同态(Endomorphism)。

注意：1)同态是树模式包含的充分条件。2)同态不是单射(Injection)函数，因此自同态的目标结点数目可能比原树模式 q 的结点数少，这意味着 q 的部分结点是冗余的。

定义 10(子查询) 对于树模式 q 和 q_s，如果 $NODES(q_s) \subseteq NODES(q)$，$C\text{-}EDGES(q_s) \subseteq C\text{-}EDGES(q)$，$D\text{-}EDGES(q_s) \subseteq D\text{-}EDGES(q)$，那么称 q_s 是 q 的子查询(Subquery)。特别地，如果 $ROOT(q) \in NODES(q_s)$，称 q_s 是 q 的有根子查询(Rooted subquery)。

定理 1 给定树模式 q，q' 是 q 的有根子查询，那么必然存在同态 $h: q' \to q$。

证明：我们构造一个映射 $h: q' \to q$，$\forall x \in NODES(q')$，$h(x)=x$。容易验证，h 满足如下 4 个条件：

1) $h(ROOT(q'))=ROOT(q')=ROOT(q)$；

2) $\forall x \in NODES(q')$，$LABEL(x)=LABEL(h(x))$；

3)如果 $(x,y) \in C\text{-}EDGES(q')$，那么 $(h(x),h(y))=(x,y) \in C\text{-}EDGES(q)$；

4)如果 $(x,y) \in D\text{-}EDGES(q')$，那么 $(h(x),h(y))=(x,y) \subseteq D\text{-}EDGES(q) \subseteq EDGES^{+}(q)$。

因此，h 满足同态映射的 4 个条件，是从 q' 到 q 的一个同态映射。□

同态映射是树模式查询包含的充分条件，定理 1 告诉我们，树模式被自身的有根子查询包含。

给定树模式 q，结点集 $S \subseteq NODES(q)$，$Q_S=\{q' \mid q'$ 是 q 的子查询，$S \subseteq NODES(q')\}$，$q_{\min}=\min(Q_S)$，则称 $q_{\min}$ 为包含 S 的 q 的最小子查询。

定理 2 给定树模式 q，q 的自同态 h_e，$S=NODES(h_e(q))$，q_m 是 q 的子查询，且 $S \subseteq NODES(q_m)$，那么 $q_m \equiv q$。

证明：根据树模式等价的定义，我们只要能构造出：1)同态映射 $h_1: q_m \to q$；2)同态映射 $h_2: q \to q_m$，即可证明 $q_m \equiv q$。

1)因为 h_e 是 q 的自同态映射，故 $h_e(ROOT(q))=ROOT(q) \in S$。又 $S \subseteq NODES(q_m)$，所以 $ROOT(q) \in NODES(q_m)$，即 q_m 是 q 的有根子查询。根据定理 1，存在同态映射 $h_1: q_m \to q$。

2)因为 $S \subseteq NODES(q_m)$，故 h_e 是从 $NODES(q)$ 到 $NODES(q_m)$ 的映射，且 q_m 是 q 的子查询，容易验证 h_e 满足同态映射的 4 个条件，是 $q \to q_m$ 的一个同态映射。

综上，$q_m \equiv q$。□

定理 2 告诉我们，可以通过计算原树模式的自同态，找到"较小"的等价树模式，这是一种新的方法。

定理 2 的一个简单实例如图 3 所示，原树模式为 q，图中结点左上方的整数是结点的唯一标号，带有标号 i 的结点记作 node-i。虚线所示是自同态映射 h_e。h_e 的映射目标结点集 $S=NODES(h_e(q))=\{$node-1, node-4, node-5, node-6$\}$。q_0 是一个包含 S 的 q 的子查询，显然 $|q_0|<|q|$，且 $q_0 \equiv q$。不难看出，q_0 是 q 的最小等价树模式。

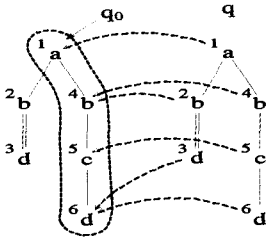

图 3 树模式 q 的自同态映射 h_e

定理 3 树模式 q 的自同态集 $\{h_i\}$，$S_i=NODES(h_i)$，q_i 是包含 S_i 的 q 的最小子查询，$q_m=\min\{q_i\}$。那么，q_m 是 $XP^{\{/,//,[],*\}}$ 片断上通过互为同态方式找到的 q 的最小等价树模式。

证明：根据定理 2，$q_m \equiv q$，这里只需证明 $q_m=\min\{q_i\}$。

假设存在通过互为同态方式获得的树模式 q_w，$q_w \equiv q$，且 $|q_w|<|q_m|$，那么存在同态映射 $h_w: q \to q_w$。对于 $XP^{\{/,//,[],*\}}$ 片断，q_w 是 q 的子查询[12]，因此 h_w 是自同态映射。一方面，$q_m=\min(q_i)$，故 $|q_m| \leqslant |q_w|$；另一方面，根据假设 $|q_w|<|q_m|$，矛盾，结论可证。□

注意：定理 3 的结论有 2 个限制条件：1)对于 $XP^{\{/,//,[],*\}}$ 片段有效；2)仅在通过互为同态映射方式获得的树模式中比较，也就是说有可能存在更小的等价树模式，但必须通过其他方式获得。

定理 3 给出了在上述限制条件下，寻找最小等价查询的框架。如果这些限制条件不能满足，那么定理 3 仍然可以用于寻找与原查询相比具有更小规模的等价子查询。根据定理 3，计算最小等价查询的框架描述如下：

1)计算树模式的自同态集；

2)比较以自同态目标结点集为基础构造的等价子查询，其中规模最小的子查询是最终结果。

对于 $XP^{\{/,//,[]\}}$ 和 $XP^{\{/,[],*\}}$，同态映射是查询包含的充要条件，因此定理 3 给出的 q_m 是 q 的最小等价查询。对于更复杂的 XPath 片段，例如 $XP^{\{/,//,[],*\}}$，定理 3 不保证获得 q_m 是 q 的最小等价查询。由于 $XP^{\{/,//,[],*\}}$ 片段的最小化问题是 coNP 完备问题，而且查询最小化只是实际查询的一个优化手段，因此我们的目标是寻找能够在多项式时间内尽可能缩小查询规模的算法。所以，定理 3 给出的框架是切实有效的。

4 高效的 XPath 查询最小化算法

以上一节的理论为基础，我们提出了一个计算 XPath 查询最小化的算法，它用到了计算同态映射的算法[7,8,11,13]，这使得本算法具有可扩展性——可以方便地用更加高效的计算同态映射算法代替当前算法。

根据定理 3 给出的框架，我们的算法分为 3 步：

(下转第 64 页)

7 Wang J, Wen J R, Lochovsky F H. Instance-based Schema Matching for Web Databases by Domain-specific Query Probing. In VLDB, 2004. 408～419
8 He B, Chang K C C, Han J. Discovering complex matchings across web query interfaces: a correlation mining approach. In KDD, 2004. 148～157
9 He B. A Holistic Paradigm for Large Scale Schema Matching. SIGMOD Record, 2004, 33(4):20～25
10 He H, Meng W, Yu C T, Wu Z. WISE-Integrator: An Automatic Integrator of Web Search Interfaces for E-Commerce. In VLDB, 2003. 357～368
11 He B, Chang K C C. Statistical Schema Matching across Web Query Interfaces. In:SIGMOD Conference, 2003. 217～228
12 Rahm E, Bernstein P A. A survey of approaches to automatic schema matching. VLDB J., 2001,10(4):334～350
13 http://www.cnnic.net.cn/download/2005/2005041401.pdf
14 Liu Q, Li S J. Word Similarity Computing Based on How-net. In: The 6th Chinese Lexical Semantics Workshop,2002

(上接第 60 页)

1. 计算树模式 q 的所有自同态及其目标结点集；

2. 计算包含 1 中每个结点集的最小子查询的结点集，同时找到其中规模最小的子查询的结点集；

3. 根据 2 的结点集构造树模式，即是最终输出。

在第 1 步中，我们记树模式 q 的自同态 h_a 的目标结点集为 S_i，$S_i = NODES(h_a(q))$，记树模式 q 的所有自同态目标结点集的集合为 $tarset(q)$，$tarset(q)=\{S_i\}$。在图 3 的例子中，树模式 q 存在 2 个自同态，一个是图中所示的自同态，记作 h_{e1}，另一个是每个结点都映射到其自身的自同态，记作 h_{e2}，它们的目标结点集分别是 $S_1=\{$node-1, node-4, node-5, node-6$\}$, $S_2=\{$node-1, node-2, node-3, node-4, node-5, node-6$\}$，所以 $tarset(q)=\{S_1, S_2\}$。

在第 2 步中，我们计算包含 S_i 的最小子查询的结点集，记为 $minisub(S_i)$，即 $minisub(S_i)=\{x|x\in S_i\}\cup\{y|(y,x)\in EDGES^+(q)\}$。同时找到所有 $minisub(S_i)$ 中规模最小的结点集，记为 $miniSet$，即 $miniSet=\min\{minisub(S_i)\}$。在图 3 的例子中，$minisub(S_1)=S_1$，$minisub(S_2)=S_2$，$miniSet=\min\{S_1, S_2\}=S_1=\{$node-1, node-4, node-5, node-6$\}$。

在第 3 步中，我们通过如下方法构造规模最小的子查询：按深度优先顺序遍历原树模式查询 q，如果当前结点不在 $miniSet$ 中，那么删除以该结点为根的子树。最终获得原树模式查询 q 的最小等价子查询。在图 3 的例子中，q_0 是 q 的最小等价子查询。

```
算法：树模式最小化算法
输入：树模式 q
输出：q 的最小等价树模式
(1)   计算 q 的所有自同态{h_ei};
(2)   计算每个自同态的目标结点集 tarset(q) = {NODES(h_ei(q))};
(3)   int miniSize = |q|;
(4)   Set miniSet = ∅;
(5)   for each S ∈ tarset(q){
(6)     minisub(S) = ∅;
(7)     for each k ∈ S{
(8)       if ( k ∉ minisub(S) )
(9)         minisub(S) = minisub(S) ∪ {k};
(10)      for each pk ∈ {pk | (pk, k) ∈ EDGES+(q)}
(11)        if (pk ∉ minisub(S))
(12)          minisub(S) = minisub(S) ∪ {pk};
(13)      if (|minisub(S)| < miniSize){
(14)        miniSize = |minisub(S)|;
(15)        miniSet = minisub(S);}}}
(16)  if (miniSize = |q|)
(17)    return q;
(18)  else{
(19)    for each x ∈ NODES(q) //深度优先顺序
(20)    if ( x ∉ miniSet)
(21)      删除以 x 为根的子树;
(22)    return q;}
```

图 4 XPath 查询最小化算法

算法如图 4 所示，第一步(行 1～2)的时间杂度主要由计算同态映射的复杂度($O(|q|^2)$)[11]决定，除此，计算目标结点集发生在已知同态映射之后，因此不增加新的时间复杂度。第二步内层循环(行 7～15)最坏情况是遍历 q 的所有结点，时间复杂度为 $|q|$；外层循环(行 5～15)的时间复杂度是 $|tarset(q)|$，这通常是一个比 $|q|$ 小的数。因此第二步的时间复杂度是 $|q|^2$。第三步(行 16～22)遍历 q 的所有结点，故时间复杂度是 $|q|$。综上，算法的时间复杂度为 $O(2|q|^2+|q|)$。

结论和展望 本文研究树模式查询最小化问题，提出了一个可扩展的最小化的框架，并给出一个有效的算法。对于不同类型的查询片段，只要有相应的计算同态映射的算法，都可以嵌入本框架，形成新的算法。本文提出的算法是高效率的，但由于存在同态映射是查询包含的充分但非必要条件，因此对于某些查询片段，该算法不能获得最小查询。考虑到查询最小化问题是 coNP 完备问题，故寻求完备性和高效性的折中是不可避免的，即使对于这些特殊的片段，我们的算法也是实用的。

参 考 文 献

1 Bray T, Paoli J, Sperberg-McQueen C M, Maler E, Microsystems S, Yergeau F, Cowan J. Extensible Markup Language (XML) 1. 1. http://www.w3.org/TR/2004/REC-xml11-20040204/
2 Berglund A, Boag S, Chamberlin D, Fernandez M F, kay M, Robie J, Simeon J. XML Path Language(XPath) 2. 0. http://www.w3.org/TR/xpath20/
3 Boag S, Chamberlin D, Fernández M F, Florescu D, Robie J, Siméon J. XQuery 1. 0: An XML Query Language. http://www.w3.org/TR/xquery
4 Amer-Yahia S, Cho S R, Lakshmanan L V S, Srivastava D. Minimization of Tree Pattern Queries. In: Proc. of the 2001 ACM SIGMOD Conf on Management of Data. Santa Barbara, California, USA, May 2001. 497～508
5 Miklau G, Suciu D. Containment and equivalence for an XPath fragment. In: Proc. 21th Symposium on Principles of Database Systems (PODS 2002), 2002. 65～76
6 Milo T, Suciu D. Index structures for path expressions. In: Proc. Database Theory - ICDT '99, 7th International Conference 1999. 277～295
7 Amer-Yahia S, Cho S R, Lakshmanan L V S, Srivastava D. Tree pattern query minimization. The VLDB Journal, 2002, 11(4): 315～331
8 Schwentick T. XPath query containment. In: ACM SIGMOD Database principles Column, ACM Press, 2004, 33(1):101～109
9 Miklau G, Suciu D. Containment and equivalence for a fragment of XPath. Journal of the ACM, 2004, 51(1): 2～45
10 Chandra A K, Merlin P M. Optimal Implementation of Conjunctive Queries in Relational Databases. In: Proc. of the 9th ACM Syrup, Boulder, Colorado, United States: Theory of Computing, 1977. 77～90
11 Ramanan P. Efficient Algorithms for Minimizing Tree Pattern Queries. In: Proc. of the 2002 ACM SIGMOD Conf. on Management of Data, Madison, Wisconsin, June 2002. 299～309
12 Flesca S, Furfaro F, Masciari E. On the minimization of XPath queries, In: Proc. of the 29th Int Conf. on Very Large Data Base, Berlin, Germany, September, 2003
13 Liao Y, Feng J, Zhang Y, Zhou L. Hidden Conditioned Homomorphism for XPath Fragment Containment. In: Proc. of the 11th International Conference on Database Systems for Advanced Applications (DASFAA'2006), Springer-Verlag, April 2006. 454～467