DOI:10.7544/issn1000-1239.2016.20150872
53(2):316-325,2016

面向数据融合的半环溯源计算方法

薛见新　申德荣　寇　月　聂铁铮　于　戈
(东北大学信息科学与工程学院　沈阳　110819)
(xuejianxin@research.neu.edu.cn)

Semiring Provenance for Data Fusion

Xue Jianxin, Shen Derong, Kou Yue, Nie Tiezheng, and Yu Ge
(*College of Information Science and Engineering, Northeastern University, Shenyang* 110819)

Abstract　As an important part of the Web data integration, Web data fusion is the quality assurance of integrated data and the precondition of accurate analysis and mining. However, being a uniform data fusion is treated as black box, which makes the fusion lack of interpretability and debuggable ability. Therefore, to describe fusion process and origin for solving the conflict, we should construct a provenance mechanism with data provenance. Data provenance describes about how data is generated and evolves with time going on, which can not only show which input tuples contribute to the data but also how they contribute. We study the semiring provenance for data fusion. Firstly, we propose an approximate iterative approach to optimal the computational process of semiring provenance. After, to speed up the convergence, we show a Newton-like approach. Recursion may make the situation complicated, we analysize the characteristic of semiring provenance and show that Kleene sequence and Newton-like sequence can convergent only after n step. And experiments show that the technologies in this paper are highly effective and feasible.

Key words　data fusion; semiring provenance; polynomial systems; derive trees; recursive queries

摘　要　数据融合是集成数据的质量保证和分析挖掘的前提条件;然而,数据融合作为一个整体对于用户来讲是一个黑盒过程,使得当前数据融合过程缺乏可解释性和可调试性.为了便于数据融合过程中有效的冲突检测和调试,需要利用数据溯源技术建立数据融合的可回溯机制.数据溯源描述了数据产生并随着时间推移而演变的整个过程,半环溯源模型作为一种经典的数据溯源表示形式,不仅能表示结果数据是由哪些数据派生的,而且还能够描述这些数据以什么方式进行派生.主要研究用于数据融合的半环溯源的计算问题.用于数据融合的半环溯源计算是一个 pay as you go 的模式,计算数据的溯源信息是一个非常耗时的过程.首先,提出一种基于 Kleene 序列的近似迭代方法,并证明了该方法与半环溯源的派生树定义的关系,从而证明了该方法的正确性.然后,提出了一种类牛顿序列,这种方法比 Kleene 序列有更好的收敛性.由于递归的引入可能会导致这 2 种迭代算法无法终止,通过分析结果元组的半环多项式溯源的特点,证明这 2 种近似算法最坏可在 n 次迭代后终止.最后,通过实验说明了本文提出的方法是可行和有效的.

关键词　数据融合;半环溯源;多项式系统;派生树;递归查询

中图法分类号　TP391

收稿日期:2015-09-25;**修回日期**:2015-11-23

基金项目:国家自然科学基金项目(61472070);国家"九七三"重点基础研究发展规划基金项目(2012CB316201)
This work was supported by the National Natural Science Foundation of China (61472070) and the National Basic Research Program of China (973 Program) (2012CB316201).

随着网络的飞速发展,Web 技术以其广泛性、交互性、快捷性和开放性等特点迅速风靡全球,并且已经渗入到社会的各个领域,网站及网页数量正以指数级飞速增长.如何准确、有效地集成海量高价值的 Web 信息,对于诸如市场情报分析、舆情分析、商业智能等分析型应用尤为重要,具有非常重要的应用价值和现实意义.

但是,由于 Web 信息发布的随意性以及信息发布者的水平差异,使得 Web 中存在大量不完整、过时、甚至错误、虚假的信息,为了保证集成的准确性,数据融合需要对多数据源的数据进行冲突解决.Web 数据融合作为一个整体对用户来讲是一个黑盒过程,这使得当前数据融合过程缺乏可解释性和可调试性.因此,为了使用户能够了解数据来源及演化过程,同时,追踪冲突数据来源以便有效地解决冲突,需要研究一种基于数据溯源的可回溯的数据融合方法.

数据溯源(data provenance)是指数据的产生、并随着时间推移而演化的整个过程的信息,可以为数据融合提供可回溯机制.但是,用于解决数据融合中数据冲突的数据溯源是一种细粒度的 pay as you go 模型,而当前细粒度的数据溯源方法要么是根据 Data lineage[1] 的方法计算数据起源,要么是预先计算所有数据的溯源形式[2].这些方法都不能满足数据融合可回溯机制的要求.

针对现在数据融合过程中透明化、融合阶段比较孤立的特点,同时为了使融合结果具备可解释性、融合过程具备可调试性,需要一种数据溯源方法来实现数据融合可回溯机制,该数据溯源方法应该:

1) 是一种细粒度的数据溯源方法,不仅能够表示数据的起源,而且还能够描述数据的生成过程.

2) 是一种快速的溯源计算方法,由于数据融合中冲突的解决是一种 pay as you go 模式,数据融合对溯源计算的实时性要求较高.

很显然基于 Datalog 查询的半环溯源模型是数据融合可回溯机制的选择.

Datalog 查询是合取查询在递归上的一种扩展.近几年 Datalog 再次成为研究热点,例如基于 Datalog 的逻辑的表述式语言的大量应用、Internet 协议与服务的设计、程序语言的分析与查询、本体的推理与查询等.Datalog 查询俨然成为数据交换和数据集成中的一种更具有表达能力的语言.然而,面向 Datalog 查询的数据溯源的研究还处于起步阶段.其中主要的研究工作是宾夕法尼亚大学提出的半环多项式溯源模型[2].

半环溯源作为一种典型的细粒度溯源方法,不仅能够追踪结果元组是由哪些数据派生得到的,还能够表示这些数据以什么方式结合派生出结果元组.结果元组的半环多项式溯源信息主要通过结果元组的派生树定义[2]计算得到.这种计算本质上是一种查询的逆操作,需要大量的访问原数据库,尤其是在分布式环境下,计代价很大.然而溯源计算是数据融合可回溯机制的前提,当前的溯源计算方法无法满足数据融合对数据溯源的需求.

本文主要研究面向数据融合的半环溯源计算问题.首先,提出一种近似序列 Kleene 序列,Kleene 序列的第 i 个值 k^i 等于结果元组的所有高度不高于 i 的派生树收益.因此,Kleene 序列就相当于按派生树的高度对结果元组的派生树进行划分.根据半环溯源模型的派生树定义,结果元组的半环溯源计算可以转换成求解 Kleene 序列.这种代数方法通过把派生树的构建转化成对相应方程组的近似求解来减少对原数据库的访问,从而大大提高了半环溯源的计算效率.

针对 Kleene 序列收敛速度较慢的问题,提出了类牛顿序列 v^i.通过扩展函数在半环域内的可导的语义给出一种类牛顿的方法.由证明可以看出类牛顿序列有比 Kleene 更好的收敛速度.

然而 Kleene 序列和类牛顿序列都是一种近似迭代方法,当 Datalog 查询出现递归时,可能会无限迭代下去.通过分析派生树的结构特性,引入 Kleene 星操作,最多派生树的高度为 n 后迭代终止,n 表示结果元组的数目.

1 相关工作

数据融合的概念最初来源于多传感器数据融合[3],是指对来自多个传感器的数据进行多级别、多方面、多层次的处理,从而产生新的有意义的信息.由于需要整合来自不同数据源的异构数据,数据集成很自然地需要进行数据融合的研究.但是对于数据融合的解释还没有一个统一的标准,其中普遍认可的是 Dong 等人在文献[4]中的解释:数据融合是指将来自不同数据源的表示同一实体的不同表象融合成一种单一的表象,并解决可能存在的数据冲突的过程.

Dong 等人[5-8]在数据集成的数据融合方面做了很多具有代表性的工作,他们借助 Web 网站的拷贝现象,通过建立数据源之间的依赖关系,进而有效地

融合多数据源的数据,并设计一个原型系统SOLOMON[9],以展示数据源之间依赖关系及数据融合过程.

数据溯源是数据管理的重要内容,特别是在Web数据集成中,由于数据源的自由性以及集成过程的多阶段性,更需要追踪数据的起源及其演化过程,以确保集成数据的质量的可解释性和可调试性.

关于数据溯源已经有了大量的研究工作.在传统的异构数据集成方法中,通常使用模式级的数据溯源方法来追踪数据在不同模式间的演化过程.由于数据融合过程中的冲突检查处理的对象是数据,因此,细粒度的数据溯源方法是本文的研究对象.元组溯源是当前数据溯源研究的热点,它在关系表上额外增加一个属性来表示标注信息,用来对元组的元数据进行管理,这些标注信息可以是概率、置信度、布尔变量等.Cui等人提出的lineage溯源[1]方法用来标注哪些元组对结果元组的生成做了贡献.Buneman等人提出的where[10]溯源方法用来表示哪些元组在一起共同构成结果元组.Green等人提出了一种更一般的标注模型——半环溯源(how-provenance[2]).

对于数据溯源的查询DBNotes[5,11]系统提出一种类似于SQL查询的语言pSQL,用来指定where溯源的传递规则.Trio[12]系统开发了新查询语言TriQL,支持不确定性和溯源查询.Karvounarakis等人[13]提出了一种基于元组、半环溯源的查询语言ProQL,可以很好地实现对溯源信息的查询,同时提供一些机制解决存储和查询的相关问题.Perm[14]是一种数据溯源原型系统,它采用非标注表示法,用查询重写规则来修改SQL查询.文献[15-16]主要研究了数据溯源的查询包含问题.文献[17]提出一种基于circuit的溯源计算方法,这种方法是面向布尔查询,主要是利用布尔电路递归地计算溯源信息,该方法可以在多项式时间构建多项式大小的溯源信息.

2 半环溯源与问题描述

在这节首先介绍Datalog查询和半环溯源的一些相关概念.

2.1 Datalog查询

一个Datalog查询通常包含一系列如下形式的规则:

$$P:-(B_1,B_2,\cdots,B_n),$$

这里$n\geqslant 0$,P表示这个Datalog规则的头部,B_1,B_2,…,B_n的表示Datalog规则的主体.本文假定读者对Datalog已有足够的了解,不再赘述.

关系在Datalog规则中由谓词表示.每个谓词拥有固定数目的参数,一个谓词和它的参数一起被称为原子.实质上,谓词就是一个返回布尔值的函数名.对于Datalog规则中的谓词可以分为扩展谓词(EDB)和内涵谓词(IDB),扩展谓词的关系存放在数据库中,内涵谓词的关系是由一个或多个Datalog规则计算出来的.

2.2 半环多项式溯源

作为一种一般的抽象溯源模型,半环溯源模型是在不完整数据库、概率数据库基础上抽象出来的,是不同溯源模型应用的一个统一框架.文献[6]给出了一个统一的数据溯源框架K关系,即提出一个半环溯源表示模型,溯源的传播是根据关系代数的演算得到的,和查询的耦合度较高.下面给出K关系的定义.

定义1. 假设有一个半环$K=(\kappa,+,\times,0,1)$,包含2个演算操作符+和×以及特异元0,1.U表示关系表上的属性集.那么U上的K关系是指一个函数$R:U.tuple\rightarrow\kappa$,其中它的支持元组集$supp(R)=\{t|R(t)\neq 0\}$.

半环溯源模型是用半环多项式来表示结果数据的生成过程.其中溯源多项式中的×操作对应关系代数中的连接操作,+操作对应关系代数中的union操作.这样就可以通过多项的结构表示结果数据是由哪些数据通过什么方式生成的.例如一个结果元组t的标注是xy^2+z+z,其中x,y,z分别表示源数据库中元组的抽象标注,这个多项式表示元组t是由3种派生路径生成,一种是通过x标注的元组与y标注的元组做连接操作生成的,别外2种派生路径是由z标注的元组直接生成.K关系本质是关系半环到标注半环的一个映射关系.

下面给出半环多项式溯源的基于派生树的定义.

定义2[2]. 给出一个可交换半环$(\kappa,+,\times,0,1)$,一个数据库实例D,查询$Q\in Datalog^{\neq}$,$Datalog^{\neq}$表示存在不等式谓词的Datalog查询,对于$Q(D)$中的任意元组t,其多项式溯源可以表示为:

$$P(t,Q,D)=\sum_{a\in A(t,Q,D)}\prod_{R_i\in body(Q)}p(Q(R_i)),\quad(1)$$

其中$A(t,Q,D)$是产生t的所有派生树集合.

定义2不仅仅说明了半环溯源的语义,同时也给出半环溯源的计算方法.

2.3 派生树

派生树是本文提出的方法的关键,本文中派生树的概念来源于上下文无关文法.

下面介绍一个在 ω-连续半环域内的关于幂级数向量 $\boldsymbol{f}$ 的派生树,如果没有特殊说明,本文中的半环都是指 ω-连续半环.派生树中每一个结点都标注为(X,j)的形式,其中 $X\in\chi$ 表示 $\boldsymbol{f}$ 中的变量,j 表示该结点依赖幂级数哪个项进行派生,它本质上表示一种项的索引.如果一个派生树 t 的结点是(X,j),映射 $\lambda(t):=(X,j)$表示该派生树与根结点的映射,$\lambda_v(t):=X$ 表示根节点的变量,$\lambda_m(t):=j$ 表示该结点按幂级数中的哪个项派生.

下面将给出关于多项式方程组的派生树定义,并介绍如何定义每个派生树的收益.

定义 3. 幂级数向量 $\boldsymbol{f}$ 的派生树和派生树的收益可递归地定义为:

如果向量 $\boldsymbol{f}$ 的任一单项式 $m_{x,j}$ 中都是常量(终止符),那么派生树 t 只包含一个被标记为(X,j)的结点,同时派生树 t 的收益 $Y(t)=m_{x,j}$.

如果单项式 $m_{x,j}=a_1X_1a_2X_2\cdots a_kX_ka_{k+1}$,$k\geqslant 1$,$t_1,t_2,\cdots,t_k$ 是 $\boldsymbol{f}$ 中的一个项,其中 $\lambda_v(t_i):=X_i$;那么以 $t_1,t_2,\cdots,t_k$ 作为子树的树 t 同样也是 $\boldsymbol{f}$ 的派生树,如果 $\lambda(t):=(X,j)$,那么 $Y(t)=a_1Y(t_1)a_2Y(t_2)\cdots a_kY(t_k)a_{k+1}$.

定义 4. 派生树 t 的高度 $h(t)$表示由根结点到叶子结点的最长的路径长度.为了表示方便,用根结点表示派生,$t^{=h}$表示根结点为 t 的高度为 h 的所有派生树集合,$t^{<h}$表示根结点为 t 的所有高度小于 h 的派生树集合.

2.4 问题描述

为了更形象地说明半环溯源计算问题,下面给出实例来详细说明.

给出 Datalog 查询如图 1(a)所示,源数据库实例抽象标注形式如图 1(b),图 1(c)表示查询结果实例.通过 Datalog 查询,能够生成如图 1(d)所示的固定点方程组.其中结元组的实例表示变量,源数据实例表示常量.最终求解的结果就是用表示常量的源数据库实例来表示结果元组的标注.

当前关于半环溯源的一些研究工作[12]是假定已知如图 1(d)所示的等式,这个等式是表示各数据库元组之间的关联关系.数据溯源的查询、传播与存储等研究都是建立在这个派生关系的基础上.生成结果元组与原数据实例之间关系的过程就是计算元组间的派生关系.因此溯源计算过程是半环溯源的研究基础.相同关系内不同元组的派生路径并不相同,因此元组的溯源计算过程以元组为处理对象.结果元组的溯源计算是一个复杂而耗时的过程,由于数据融合中对时效性要求较高,因此需要快速而高效的溯源计算方法.

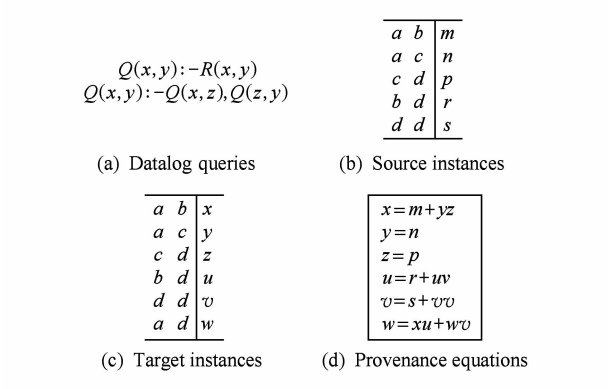


Fig. 1 Provenance for result tuples.

图 1 结果元组溯源实例

当前 Web 数据更新与演化比较频繁.而当数据经过多次演化后,多项式溯源的项会指数倍增加,这将导致溯源信息量过大,使得溯源计算过程更加复杂,而快速的计算溯源信息就成为一种挑战.

递归的引入会使得溯源计算过程更加复杂化.递归导致一些元组会重复地派生自身,结果元组的溯源表达式是形式幂级数.面向递归查询的溯源信息计算代价很大,同时溯源信息的存储将会占用更多空间.解决递归查询引起的无穷溯源问题是半环多项式溯源计算专有的问题.

3 Datalog 查询与多项式方程组

第 2 节给出了结果元组的半环多项式溯源的定义,显然这个定义不是一个有效的计算方法.因此,这种半环多项式的派生树定义只可以用来作为理论证明.从 Datalog 查询评估的角度来看,会存在一个固定点理论.正如溯源信息的 Datalog 查询一样,会存在一种与派生树定义等价的固定点语义,这种定义具有更好的可计算性.

根据 Datalog 查询评估的特点,可知结果元组可能会由其他 IDB 元组派生得到,因此,为了方便表示,对每一个结果元组使用新的变量来表示.因此,通过 Datalog 规则可以构建结果元组与所有其他元组之间的一个等价关系.

这种等价关系的构建可以在查询执行时构建.但是,一方面需要重写查询机制;另一方面查询执行过程并不能完全反映结果元组的派生路径,比如递归,因为递归查询的终止条件是两次的查询结果相同;而且,数据融合过程中的冲突检测是一个 pay as you go 模型,只有需要时才会计算相应数据的溯源信息,这种在查询过程中预先计算所有数据的溯源信息方法会大大降低查询的执行效率,也会计算很多不需要的溯源信息并占用大量的存储空间.本节提出一种查询逆操作的方程组构建方法.

定理 1. 利用 Datalog 查询构建结果元组的多项式方程组是一个 NP-hard 问题.

证明. 多项式方程组的构建主要利用 Datalog 查询找出结果元组与其他元组之间的等价关系,如果 Datalog 规则是合取式的话,那么这个问题可规约为 SAT 问题,显然是一个 NP-complete 问题.所以,利用 Datalog 查询构建结果元组的多项式方程组至少是 NP-complete 问题. 证毕.

对于一个 NP-hard 问题要从算法的角度优化该问题是比较困难的,但是可以从减少输入的角度来优化该问题.

为了简化概念,假定只存在一个 EDB 谓词和一个 IDB 谓词.对于给定的有限的 EDB 谓词 k-关系可以按如下方法(算法 1)构建固定点方程组:

算法 1. GenerateEquations.

输入:查询 Q,IDB 元组 I,EDB 元组 E;

输出:固定点方程组.

① initial;

② $EQ \leftarrow \varnothing$;

③ for each $t \in I \cup E$ do

④ $X_t \leftarrow t.ids$;

⑤ for each $i \in I$ do

⑥ $m \leftarrow mathch(Q, i, E)$;

⑦ for $n=1$, $n<number(m)+1$, n++;

⑧ $X_i = X_i + m_n$;

⑨ $EQ \leftarrow EQ \cup X_i$;

⑩ return EQ.

算法 1 主要致力于通过 Datalog 查询规则构建结果元组与其他元组之间的等价关系.首先对于每个 EDB 元组和 IDB 元组都赋予一个变量来唯一表示这个元组(行③④).EDB 元组变量在固定点方程中表示常量,而 IDB 元组变量在固定点方程中表示变量.算法 1 的关键就是找出 IDB 变量与其他所有变量之间的等价关系.这种等价关系就是根据 Datalog 查询找出所有能派生出 IDB 元组的 EDB 元组的赋值.由于这个问题超出本文的范围,在另一篇文章中将有详细的解决方法.

4 基于 Kleene 迭代的半环溯源计算方法

第 3 节介绍了结果元组半环多项式溯源计算到固定点方程组的转换.那么是不是有可能有一种对方程组的处理方法能够有效地计算结果元组的半环多项式溯源?对固定点方程组的求解显然要比根据结果元组的派生树定义直接构建派生树要有效得多,因为可以避免大量对原数据库的访问.

定理 2. 给定源数据库实例 S、目标数据库实例 T、Datalog 查询规则 q.那么,由算法 1 构建的固定点方程组的解等于结果元组的半环多项式溯源.

证明. 根据 Kleene 定理可知,在 ω-连续半环内求解多项式方程组 $\boldsymbol{X}=\boldsymbol{f}(\boldsymbol{X})$,存在 Kleene 近似序列,使得方程组 $\boldsymbol{X}=\boldsymbol{f}(\boldsymbol{X})$ 的解等于这个序列的上确界,而 Kleene 序列可以迭代地表示为 $\boldsymbol{k}^0=\boldsymbol{0}$,$\boldsymbol{k}^{i+1}=\boldsymbol{f}(\boldsymbol{k}^i)$.显然 Kleene 序列可以看成是一种迭代的方程组求解方法.为了证明定理,只需要证明 Kleene 迭代与结果元组的派生树的关系. 证毕.

引理 1. 固定点方程组的第 i 个 Kleene 迭代 $\boldsymbol{k}^i$ 等于高度不大于 i 的所有变量的派生树的收益之和.

证明. 可以通过归纳法来证明.下面的 H^i 表示度不高于 i 的所有派生树集合.

当 $i=1$ 时,$\boldsymbol{k}^0=\boldsymbol{0}$,$H^0=\boldsymbol{0}$;假定当 $i=h$ 时,$\boldsymbol{k}^h=Y(H^h)$;那么,

$$\boldsymbol{k}^{h+1}=\boldsymbol{f}(\boldsymbol{k}^h)=\sum_{j\in J} m_{X,j}(\boldsymbol{k}^h),$$

其中,

$$\begin{gathered} m_{X,j}=a_1X_1a_2X_2\cdots a_kX_ka_{k+1}= \\ a_1\boldsymbol{k}^h_{X_1}a_2\boldsymbol{k}^h_{X_2}\cdots a_k\boldsymbol{k}^h_{X_k}a_{k+1}= \\ a_1Y(H^h_{X_1})\ a_2Y(H^h_{X_2})\cdots a_k(H^h_{X_k})a_{k+1}= \\ a_1Y(t_1)\ a_2Y(t_2)\cdots a_k\ Y(t_k)a_{k+1}= \\ Y(t)\ (h(t)\leqslant h+1,\lambda(t)=(X,j)). \end{gathered}$$

很显然,$\sum_{j\in J} m_{X,j}(\boldsymbol{k}^h)=Y(H^h)$. 证毕.

由于固定点方程组的变量表示的是 IDB 元组的标注,因此,Kleene 序列就相当于按派生树的高度来构建结果元组的派生树.根据定义 2,可知由算法 1 生成的于固定点方程组的解就是结果元组的半环多项式溯源信息.

定理 2 的证明不仅说明了半环多项式溯源的计算可以转化成一种数学计算,而且还给出了一种近似迭代的半环溯源计算方法.这种方法可以避免大量的访问数据,因此,能有效地提高半环多项式标注的计算过程.

5 类牛顿迭代的半环溯源计算

尽管 Kleene 迭代方法是一种通用方法.但是对于递归查询来说可能无法终止,并且收敛速度较慢.

牛顿迭代是数值域内求解非线性方程组的一种经典方法,它本质上是将非线性方程逐步转换成线性方程来求解,是平方收敛的.那么在半环域内是不是能够有一种迭代方法能快速地收敛.

把牛顿迭代的思想扩展到半环域内,由算法 1 得到的固定点方程组的一般形式为:

$$\begin{aligned} x_1 &= f_1(x_1,x_2,\cdots,x_n);\\ x_2 &= f_2(x_2,x_2,\cdots,x_n);\\ &\vdots\\ x_n &= f_n(x_1,x_2,\cdots,x_n). \end{aligned}$$

这种方程组可以表示成向量的形式 $\boldsymbol{X}=\boldsymbol{f}(\boldsymbol{X})$.计算 $\boldsymbol{X}=\boldsymbol{f}(\boldsymbol{X})$的最小解就相当于计算 $\boldsymbol{g}(\boldsymbol{X})=\boldsymbol{0}$,其中 $\boldsymbol{g}(\boldsymbol{X})=\boldsymbol{f}(\boldsymbol{X})-\boldsymbol{X}$.

利用牛顿迭代求解方程组,给出可导函数 g_1,$g_2,\cdots,g_n:R_n\to R$,求解半环多项式溯源方法就是计算方程组 $\boldsymbol{g}(\boldsymbol{X})=\boldsymbol{0}$ 的解,其中 $\boldsymbol{g}=(g_1,g_2,\cdots,g_n)$;存在这样一个序列 $\boldsymbol{v}^{i+1}=\boldsymbol{v}^i+\boldsymbol{\Delta}^i$,其中 $\boldsymbol{\Delta}^i$ 是如下线性方程组的解:

$$\begin{aligned} D_{g_1}|_{v_i}(\boldsymbol{X})+g_1(\boldsymbol{v}^i) &= 0;\\ D_{g_2}|_{v_i}(\boldsymbol{X})+g_2(\boldsymbol{v}^i) &= 0;\\ &\vdots\\ D_{g_n}|_{v_i}(\boldsymbol{X})+g_n(\boldsymbol{v}^i) &= 0. \end{aligned}$$

其中 $D_{g_j}|_{v_i}(\boldsymbol{X})$是函数 g_j 在 $\boldsymbol{v}^i$ 的导数,也就是 g_j 的过点$(\boldsymbol{v}^i,g_j(v^i))$的切超平面.函数 $\boldsymbol{g}$ 在点 v 上是可导的,那么对于每一个变量 $\boldsymbol{X}\in\chi$ 都会存在一个函数 $D_{\boldsymbol{X}}\boldsymbol{g}|_v$ 满足 $D\boldsymbol{g}|_v=\sum\limits_{\boldsymbol{X}\in\chi}D_{\boldsymbol{X}}\boldsymbol{g}|_v$,这些函数类似于数值函数里的偏导数.

可以用向量的形式表示上面方程组,$D\boldsymbol{g}|_{v_i}(\boldsymbol{X})+\boldsymbol{g}(\boldsymbol{v}^i)=\boldsymbol{0}$,对于固定点方程组的求解问题,可以看成如下问题:

由 $\boldsymbol{v}^0$ 开始,按 $\boldsymbol{v}^{i+1}=\boldsymbol{v}^i+\boldsymbol{\Delta}^i$ 迭代地计算,其中 $\boldsymbol{\Delta}^i$ 是线性固定点方程 $D\boldsymbol{f}|_{v_i}(\boldsymbol{X})+\boldsymbol{f}(\boldsymbol{v}^i)-\boldsymbol{v}^i=\boldsymbol{X}$ 的最小解.这个序列的极值就是原方程组的解.

然而牛顿迭代到半环域内的没有可导函数,这里扩展了半环域内的函数的可导性.

定义 5. $\boldsymbol{f}$ 是半环域 S 内的形式幂级数,其中 $\boldsymbol{X}\in\chi$ 是其中的变量.关于变量 $\boldsymbol{X}$ 的在 v 点的微分函数 $D_X\boldsymbol{f}|_v(\boldsymbol{X}):v\to S$ 定义如下:

$$D_{\boldsymbol{X}}\boldsymbol{f}|_v(\boldsymbol{X})\begin{cases} 0,\ \text{if } \boldsymbol{f}\in S \text{ or } \boldsymbol{f}\in\chi-\{\boldsymbol{X}\};\\ \boldsymbol{b}_{\boldsymbol{X}},\ \text{if } \boldsymbol{f}=\boldsymbol{X};\\ D_{\boldsymbol{X}}\boldsymbol{g}|_v(\boldsymbol{b})\times\boldsymbol{h}(v)+\boldsymbol{g}(v)\times D_{\boldsymbol{X}}\boldsymbol{h}|_v(\boldsymbol{b}),\\ \quad\text{if } \boldsymbol{f}=\boldsymbol{g}\times\boldsymbol{h};\\ \sum\limits_{i\in I}D_{\boldsymbol{X}}\boldsymbol{f}_i|_v(\boldsymbol{b}),\ \text{if } \boldsymbol{f}=\sum\limits_{i\in I}\boldsymbol{f}_i. \end{cases}$$

下面定义函数 $\boldsymbol{f}$ 在 v 点的函数

$$D\boldsymbol{f}|_v:=\sum_{\boldsymbol{X}\in\chi}D_{\boldsymbol{X}}\boldsymbol{f}|_v,$$

因此,向量 $\boldsymbol{f}$ 在 v 可以被定义为函数 $D\boldsymbol{f}|_v$:

$$(D\boldsymbol{f}|_v(\boldsymbol{b}))_{\boldsymbol{X}}:=D\boldsymbol{f}_{\boldsymbol{X}}|_v(\boldsymbol{b}).$$

由于在半环域内没有减法,$D\boldsymbol{f}|_{v_i}(\boldsymbol{X})+\boldsymbol{f}(\boldsymbol{v}^i)-\boldsymbol{v}^i=\boldsymbol{X}$ 可变成

$$D\boldsymbol{f}|_{v_i}(\boldsymbol{X})+\boldsymbol{\delta}^i=\boldsymbol{X},$$

其中 $\boldsymbol{\delta}^i$ 是满足方程 $\boldsymbol{f}(\boldsymbol{v}^i)=\boldsymbol{v}^i+\boldsymbol{\delta}^i$ 的取值.

定义 6. $\boldsymbol{f}$ 是一组形式幂级数的向量,第 i 个类牛顿近似 $\boldsymbol{v}^i$ 可以被归纳地定义为:

$$\boldsymbol{v}^0=\boldsymbol{f}(\boldsymbol{0})\ \text{和}\ \boldsymbol{v}^{i+1}=\boldsymbol{v}^i+\boldsymbol{\Delta}^i,$$

其中 $\boldsymbol{\Delta}^i$ 是方程 $D\boldsymbol{f}|_{v_i}(\boldsymbol{X})+\boldsymbol{\delta}^i=\boldsymbol{X}$ 的最小解,$\boldsymbol{\delta}^i$ 是满足方程 $\boldsymbol{f}(\boldsymbol{v}^i)=\boldsymbol{v}^i+\boldsymbol{\delta}^i$ 的任意取值,序列 $\boldsymbol{v}^i$ 被称为类牛顿序列.

给出近似迭代方法类牛顿序列,下面给出类牛顿序列与 Klenne 序列的关系.

引理 2. $\boldsymbol{f}:V\to V$ 是一组形式幂级数的向量,其中 $\boldsymbol{u},\boldsymbol{w}\in V$,那么 $D\boldsymbol{f}|_{\boldsymbol{u}}(\boldsymbol{X})+\boldsymbol{w}=\boldsymbol{X}$;的最小解是 $D\boldsymbol{f}|*_{\boldsymbol{u}}(\boldsymbol{w})$.相似地,类牛顿序列可以被定义为 $\boldsymbol{w}^0=\boldsymbol{f}(\boldsymbol{0})$ 和 $\boldsymbol{w}^{i+1}=\boldsymbol{w}^i+D\boldsymbol{f}|*_{v_i}(\boldsymbol{\delta}^i)$.

证明. $*$操作是 Kleene 星操作,$D\boldsymbol{f}|_{\boldsymbol{u}}(\boldsymbol{X})$的 Kleene 星操作定义为 $D\boldsymbol{f}|*_{\boldsymbol{u}}(\boldsymbol{v})=\sum\limits_{i\in N}D\boldsymbol{f}|_{\boldsymbol{u}}^i(\boldsymbol{w})$,那么$D\boldsymbol{f}|*_{\boldsymbol{u}}(\boldsymbol{v})=\boldsymbol{w}+D\boldsymbol{f}|*_{\boldsymbol{u}}(D\boldsymbol{f}|*_{\boldsymbol{u}}(\boldsymbol{w}))$,很显然 $D\boldsymbol{f}|_{\boldsymbol{u}}(\boldsymbol{X})+\boldsymbol{w}=\boldsymbol{X}$ 的最小解是 $D\boldsymbol{f}|*_{\boldsymbol{u}}(\boldsymbol{w})$.

引理 3 是 Taylor's 定理在半环域内的扩展.

引理 3. $\boldsymbol{f}:V\to V$ 是一组形式幂级数的向量,其中 $\boldsymbol{u},\boldsymbol{v}\in V$,那么:

$$\boldsymbol{f}(\boldsymbol{u})+D\boldsymbol{f}|_{\boldsymbol{u}}(\boldsymbol{v})\leqslant\boldsymbol{f}(\boldsymbol{u}+\boldsymbol{v})\leqslant\boldsymbol{f}(\boldsymbol{u})+D\boldsymbol{f}|_{\boldsymbol{u}+\boldsymbol{v}}(\boldsymbol{v}).$$

证明. 对于由于 $\boldsymbol{f}$ 一组形式幂级数的向量,可以假定 ρ 表示向量 $\boldsymbol{f}$ 中的一个元素,那么 ρ 一般形式

$\rho=g\times\chi\times a$,其中,g 表示单项式,χ 表示变量,a 表示常量.

$$\rho(\boldsymbol{u}+\boldsymbol{v})=g(\boldsymbol{u}+\boldsymbol{v})\times(\boldsymbol{u}_\chi+\boldsymbol{v}_\chi)\times a\geqslant$$
$$(g(\boldsymbol{u})+Dg\mid_{\boldsymbol{u}}(\boldsymbol{v}))\times(\boldsymbol{u}_\chi+\boldsymbol{v}_\chi)\times a=$$
$$g(\boldsymbol{u})\times\boldsymbol{u}_\chi\times a+g(\boldsymbol{u})\times\boldsymbol{v}_\chi\times a+Dg\mid_{\boldsymbol{u}}(\boldsymbol{v})\times$$
$$(\boldsymbol{u}_\chi+\boldsymbol{v}_\chi)\times a\geqslant\rho(\boldsymbol{u})+g(\boldsymbol{u})\times$$
$$\boldsymbol{v}_\chi\times a+Dg\mid_{\boldsymbol{u}}(\boldsymbol{v})\times\boldsymbol{u}_\chi\times a=\rho(\boldsymbol{u})+D\rho\mid_{\boldsymbol{u}}(\boldsymbol{v});$$
$$\rho(\boldsymbol{u}+\boldsymbol{v})=g(\boldsymbol{u}+\boldsymbol{v})\times(\boldsymbol{u}_\chi+\boldsymbol{v}_\chi)\times a\leqslant$$
$$(g(\boldsymbol{u})+Dg\mid_{\boldsymbol{u}+\boldsymbol{v}}(\boldsymbol{v}))\times(\boldsymbol{u}_\chi+\boldsymbol{v}_\chi)\times a=$$
$$g(\boldsymbol{u})\times\boldsymbol{u}_\chi\times a+g(\boldsymbol{u})\times\boldsymbol{v}_\chi\times a+Dg\mid_{\boldsymbol{u}+\boldsymbol{v}}(\boldsymbol{v})\times$$
$$(\boldsymbol{u}_\chi+\boldsymbol{v}_\chi)\times a\leqslant\rho(\boldsymbol{u})+$$
$$g(\boldsymbol{u}+\boldsymbol{v})\times\boldsymbol{v}_\chi\times a+Dg\mid_{\boldsymbol{u}+\boldsymbol{v}}(\boldsymbol{v})\times$$
$$(\boldsymbol{u}_\chi+\boldsymbol{v}_\chi)\times a=\rho(\boldsymbol{u})+D\rho\mid_{\boldsymbol{u}+\boldsymbol{v}}(\boldsymbol{v}).$$

由此可证 $\boldsymbol{f}(\boldsymbol{u})+D\boldsymbol{f}\mid_{\boldsymbol{u}}(\boldsymbol{v})\leqslant\boldsymbol{f}(\boldsymbol{u}+\boldsymbol{v})\leqslant\boldsymbol{f}(\boldsymbol{u})+D\boldsymbol{f}\mid_{\boldsymbol{u}+\boldsymbol{v}}(\boldsymbol{v})$. 证毕.

引理 4. 如果 $f(\boldsymbol{x})\geqslant\boldsymbol{x}$,那么对于所有的 d 来说,会存在一个向量 $\boldsymbol{e}^{(d)}(\boldsymbol{x})$,使得

$$\boldsymbol{f}^d(\boldsymbol{x})+\boldsymbol{e}^{(d)}(\boldsymbol{x})=\boldsymbol{f}^{d+1}(\boldsymbol{x}),$$
$$\boldsymbol{e}^{(d)}(\boldsymbol{x})\geqslant D\boldsymbol{f}\mid_{fd-1}(D\boldsymbol{f}\mid_{fd-2}(\cdots D\boldsymbol{f}\mid_{\boldsymbol{x}}$$
$$(\boldsymbol{e}^{(0)}(\boldsymbol{x}))))\geqslant D\boldsymbol{f}\mid_{\boldsymbol{x}}^{d}(\boldsymbol{e}^{(0)}(\boldsymbol{x})).$$

证明. 通过归纳法证明,当上标为 0 时显然是满足的,假定上标为 d 时也是满足的,那么,

$$\boldsymbol{f}^{d+1}(\boldsymbol{x})=\boldsymbol{f}(\boldsymbol{f}^d(\boldsymbol{x})+\boldsymbol{e}^{(d)}(\boldsymbol{x}))\geqslant$$
$$\boldsymbol{f}^{d+1}(\boldsymbol{x})+D\boldsymbol{f}\mid_{fd(\boldsymbol{x})}(\boldsymbol{e}^{(d)}(\boldsymbol{x})\geqslant$$
$$\boldsymbol{f}^{d+1}(\boldsymbol{x})+D\boldsymbol{f}\mid_{fd-1}(D\boldsymbol{f}\mid_{fd-2}(\cdots D\boldsymbol{f}\mid_{\boldsymbol{x}}(\boldsymbol{e}^{(0)}(\boldsymbol{x})))).$$

因此会存在 $\boldsymbol{e}^{(d+1)}(\boldsymbol{x})\geqslant D\boldsymbol{f}\mid_{fd-1}(D\boldsymbol{f}\mid_{fd-2}(\cdots D\boldsymbol{f}\mid_{\boldsymbol{x}}(\boldsymbol{e}^{(0)}(\boldsymbol{x}))))$,由于 $D\boldsymbol{f}\mid y$ 是单调的,因此引理可证. 证毕.

定理 3. $\boldsymbol{f}$ 是一组形式幂级数的向量,那么只会存在一个类牛顿近似 $\boldsymbol{v}^i$,满足如下性质:

$$\boldsymbol{k}^i\leqslant\boldsymbol{v}^i\leqslant\boldsymbol{f}(\boldsymbol{v}^i)\leqslant\boldsymbol{v}^i+1\leqslant\mu f=\sup\boldsymbol{k}^j.$$

证明. 首先证明对于所有的 i 来说会存在一个 $\boldsymbol{\delta}^i$,满足 $\boldsymbol{k}^i\leqslant\boldsymbol{v}^i\leqslant\boldsymbol{f}(\boldsymbol{v}^i)$. 利用归纳法证明这个结论:

当 $i=0$ 时,结果显然是满足的. 那么假定 i 取任意大于 0 的整数时也满足,即 $\boldsymbol{k}^i\leqslant\boldsymbol{v}^i\leqslant\boldsymbol{f}(\boldsymbol{v}^i)$.

$$\boldsymbol{k}^{i+1}=\boldsymbol{f}(\boldsymbol{k}^i)\leqslant\boldsymbol{f}(\boldsymbol{v}^i)=$$
$$\boldsymbol{v}^i+\boldsymbol{\delta}^i\leqslant\boldsymbol{v}^i+D\boldsymbol{f}\mid*_{v_i}(\boldsymbol{\delta}^i)=\boldsymbol{v}^{i+1}=$$
$$\boldsymbol{v}^i+\boldsymbol{\delta}^i+D\boldsymbol{f}\mid*_{v_i}(D\boldsymbol{f}\mid*_{v_i}(\boldsymbol{v}))=$$
$$\boldsymbol{f}(\boldsymbol{v}^i)+D\boldsymbol{f}\mid*_{v_i}(D\boldsymbol{f}\mid*_{v_i}(\boldsymbol{v}))\leqslant$$
$$\boldsymbol{f}(\boldsymbol{v}^i+D\boldsymbol{f}\mid*_{v_i}(\boldsymbol{\delta}^i))=\boldsymbol{f}(\boldsymbol{v}^{i+1}).$$

由于 $\boldsymbol{v}^{i+1}\leqslant\boldsymbol{f}(\boldsymbol{v}^{i+1})$,那么会存在 $\boldsymbol{\delta}^{i+1}$,使得 $\boldsymbol{v}^{i+1}+\boldsymbol{\delta}^{i+1}\leqslant\boldsymbol{f}(\boldsymbol{v}^{i+1})$. 接下来证明 $\boldsymbol{f}(\boldsymbol{v}^i)\leqslant\boldsymbol{v}^{i+1}$:

$$\boldsymbol{f}(\boldsymbol{v}^i)=\boldsymbol{v}^i+\boldsymbol{\delta}^i\leqslant\boldsymbol{v}^i+D\boldsymbol{f}\mid*_{v_i}(\boldsymbol{\delta}^i)=\boldsymbol{v}^{i+1}.$$

接下来需要证明 $\boldsymbol{v}^i\leqslant\mu f$.

当迭代次数为 0 时,显然是满足的,假定当迭代次数为 i 时也满足:

$$\boldsymbol{v}^{i+1}=\boldsymbol{v}^i+D\boldsymbol{f}\mid*_{v_i}(\boldsymbol{\delta}^i)=$$
$$\boldsymbol{v}^i+\sum_{d\in n}D\boldsymbol{f}\mid_{v_i}^{d}(\boldsymbol{\delta}^i)\leqslant$$
$$\boldsymbol{v}^i+\sum_{d\in n}\boldsymbol{e}^{(d)}(\boldsymbol{v}^i)=\sup_{d\in n}\boldsymbol{f}^d(\boldsymbol{v}^i)\leqslant\mu f.$$ 证毕.

定理 3 的证明说明了类牛顿序列比 Kleene 序列有更好的收敛性,也就是能更快地计算出半环多项式溯源.

由于 Kleene 序列和类牛顿序列都是一种迭代方法,而当 Datalog 查询出现递归时迭代可能是无法终止的.

定理 4. Kleene 序列和类牛顿序列最多可经过 n 步终止,其中 n 表示结果元组的数目.

证明. 由于第 n 个 Kleene 序列表示所有高度不大于 n 的派生树集合. 定理就相当于证明派生树的高度不需要大于 n 就可以计算出结果元组的半环多项式溯源. 证毕.

6 实验结果与分析

本节给出了基于 Kleene 迭代方法和类牛顿迭代方法的结果元组半环多项式溯源计算的性能测试结果,并与基于派生树构建的半环多项式标注计算方法进行比较.

采用 TPC-H Benchmark 的数据集来进行测试. TPC-H Benchmark 是通过数据库查询性能测试的实验平台,本文在 TPC-H Benchmark 测试中选用了查询结果中不包含聚类操作的查询 Q_2,Q_{11},Q_{15} 和 Q_{20}.

此次实验使用的数据库是 sql sever 2008 数据库,开发工具使用 vs2010,系统实现的语言为 C#.

6.1 不同数据大小的溯源计算比较

首先考虑的是输入数据大小对各算法的执行时间的影响. 选择查询 Q_2,Q_{11},Q_{15} 和 Q_{20} 来度量面向不同的查询结果元组的半环多项式溯源计算过程的执行时间.

Tradition 方法指的是传统的基于派生树构建的方法,Kleene 是指 Kleene 近似迭代方法,Newton 是指类牛顿迭代方法.

由图 2 中可以看出传统的基于派生树构建的半环多项式溯源计算方法随着元组数的增长计算代价

增长较快,而 Kleene 近似方法和类牛顿近似方法的计算代价随着数据的增长近似线性,可见 Kleene 近似方法与类牛顿近似方法有很好的计算效率,同时也有很好的可扩展性.

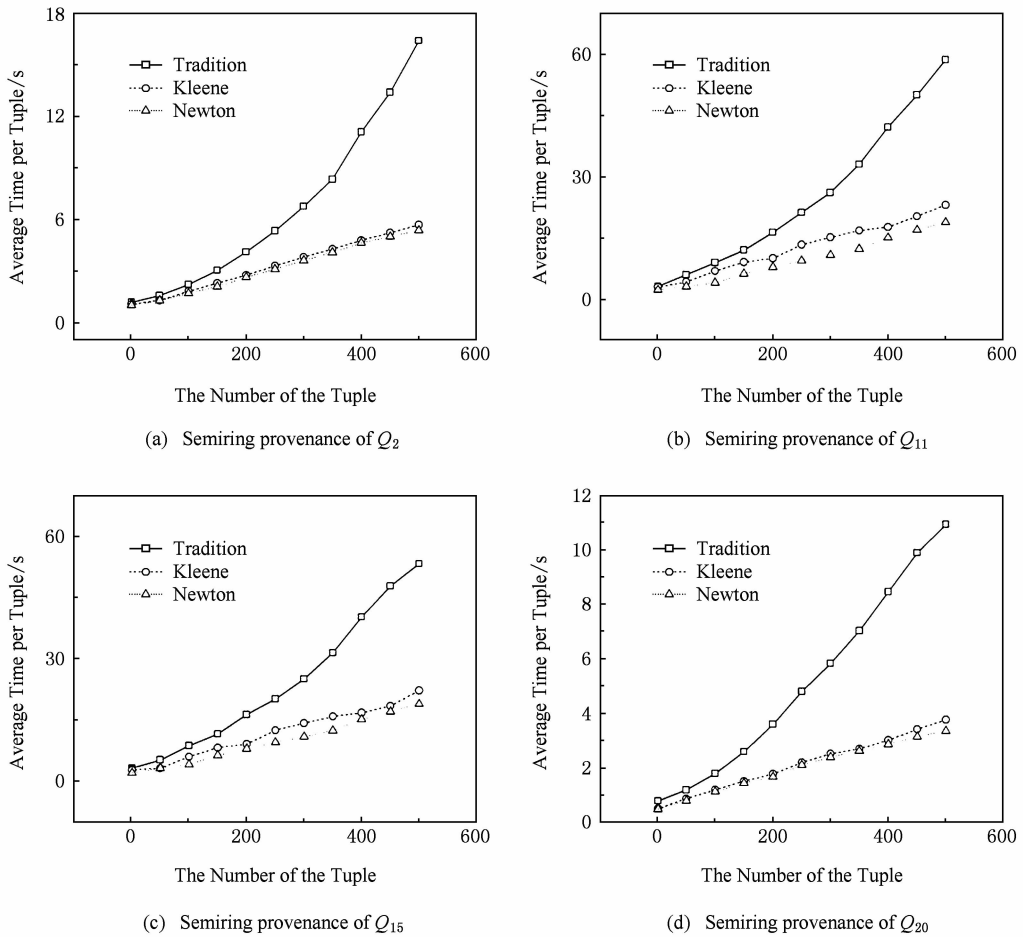


Fig. 2 Comparison of semiring provenance with different queries.

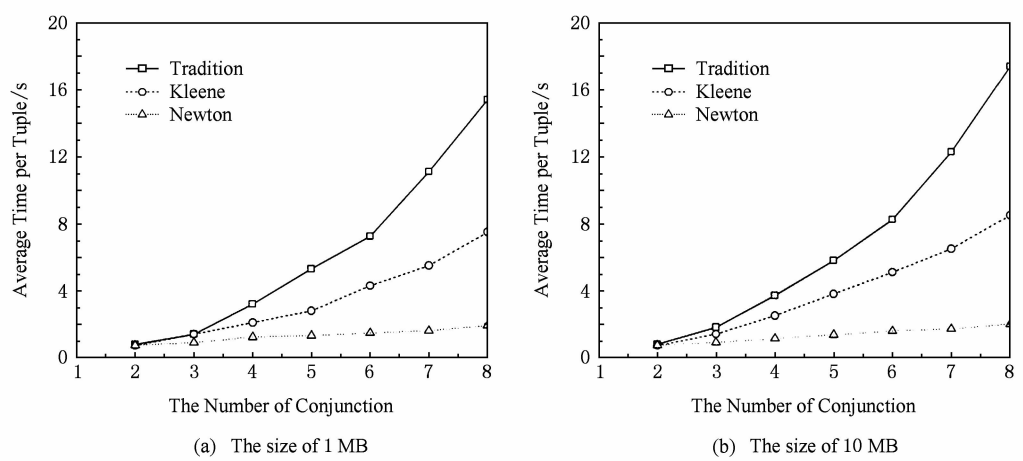
图 2 不同查询的半环多项式溯源计算对比

6.2 合取谓词不同对溯源计算的影响

半环多项式溯源主要是根据 Datalog 查询找出满足查询的所有元组的组合,本小节测试溯源计算方法面对不同长度的 Datalog 规则的计算效率.

图 3 所示为数据大小为 1 MB,10 MB,100 MB,500 MB 的情况下合取式大小由 2~8 的计算代价.

通过图 3 可以看出传统的基于派生树的构建方法代价要远远高于基于 Kleene 迭代和类牛顿迭代的方法.而随着合取式数目的增加类牛顿迭代表现了更好的性能.



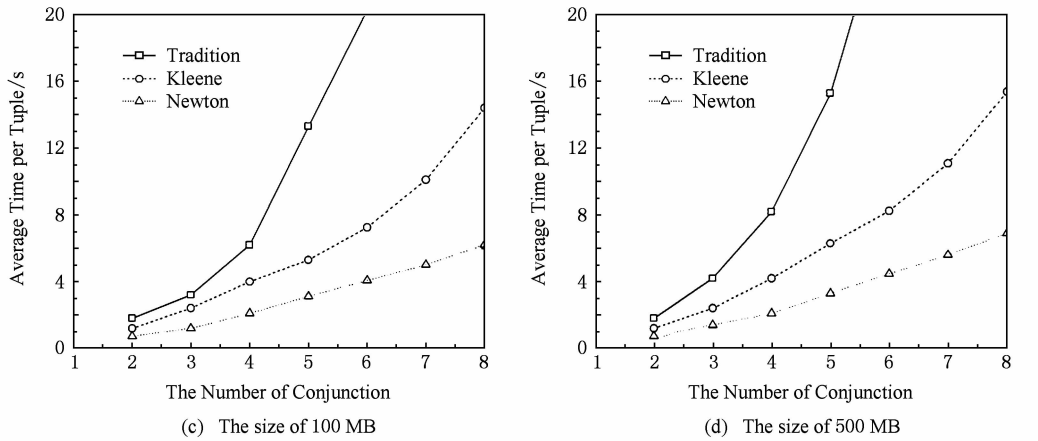


Fig. 3 Comparison of semiring provenance with different size of queries.

图 3 不同查询长度的半环多项式溯源计算对比

6.3 不同演化版本的半环溯源计算分析

针对 TPCH 数据集构建不同版本的数据,每个版本都可以用实化视图表示. 图 4 中对源数据大小为 10 MB 和 100 MB 两种情况进行了分析比较. 半环多项式溯源的计算代价随着版本的增加快速增加,很显然传统的方法增长很快,而随着版本数的增加,类牛顿迭代表现了很好的计算性能和良好的可扩展性.

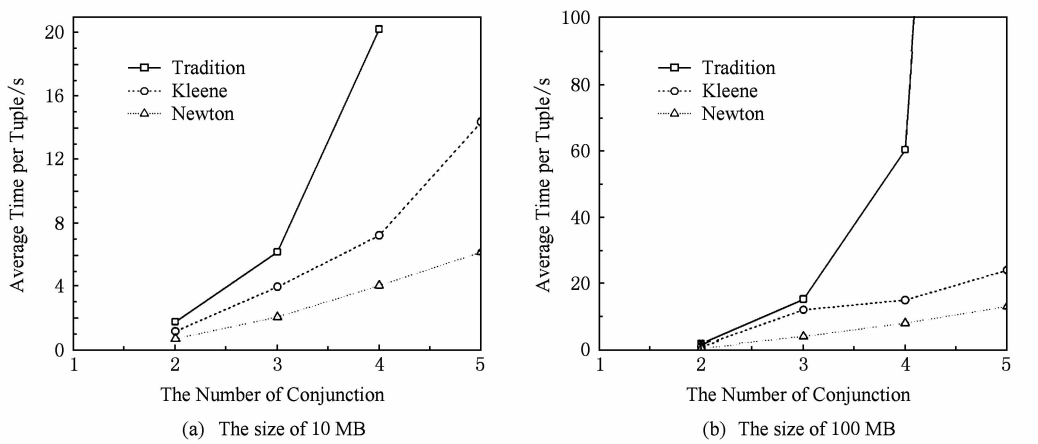


Fig. 4 Comparison of semiring provenance with different versions.

图 4 不同演化版本下的半环多项式溯源比较

7 结束语

针对数据融合过程缺乏可解释性和可调试性,同时为了便于数据融合过程中有效的冲突检测和调试,本文提出了一种基于数据融合的可回溯机制. 本文方法致力于研究快速的溯源计算方法,以应对数据融合过程的可回溯机制对实时性的要求. 首先提出一种基于 Kleene 序列的近似迭代方法,该方法通过把对数据处理转换对半环多项式的求解来避免对数据库的大量访问;然后提出一种类牛顿迭代方法通过加速收敛来提高溯源计算效率. 通过分析,本文提出的 2 种迭代方法均收敛. 实验结果表明,相对于传统的基于派生树的溯源计算方法,本文提出的 2 种方法有正好的计算效率,同时随着数据量增长,类牛顿迭代方法有更好的可扩展性.

参 考 文 献

[1] Cui Y, Widom J, Wiener J L. Tracing the lineage of view data in a warehousing environment [J]. ACM Trans on Database Systems, 2000, 25(2): 179-227

[2] Green T J, Karvounarakis G, Tannen V. Provenance semirings [C] //Proc of the 27th ACM SIGMOD Int Conf on Management of Data Symp on Principles of Database Systems. New York: ACM, 2007: 31-40

[3] Llinas J, Hall D L. An introduction to multi-sensor data fusion [C] //Proc of ISCAS'98. Piscataway, NJ: IEEE, 1998: 537-540

[4] Dong X L, Naumann F. Data fusion—Resolving data conflicts for integration [J]. Proceedings of the VLDB Endowment, 2009, 2(2): 1654-1655

[5] Dong X L, Gabrilovich E, Heitz G, et al. From data fusion to knowledge fusion [J]. Proceedings of the VLDB Endowment, 2014, 7(10): 881-892

[6] Dong X L, Berti-Equille L, Srivastava D. Data fusion: Resolving conflicts from mutiple sources [C] //Proc of the WAIM2013. Berlin: Springer, 2013: 64-76

[7] Li Xian, Dong X L, Lyons K, et al. Scaling up copy detection [C] //Proc of the 31st IEEE Int Conf on Data Engineering. Piscataway, NJ: IEEE, 89-100

[8] Xuan Liu, Xin Luna Dong, Beng Chin Ooi, et al. Online data fusion [J]. Proceedings of the VLDB Endowment, 2011, 4 (11): 932-943

[9] Dong X L, Berti-Equille L, Hu Yifan, et al. Solomon: Seeking the truth via copying detection [J]. Proceedings of the VLDB Endowment, 2010, 3(12): 3-3

[10] Buneman P, Khanna S, Tan W C. Why and where: A characterization of data provenance [C]//Proc of the 8th Int Conf on Data Theory. Berlin: Springer, 2001: 316-330

[11] Chiticariu L, Tan W C, Vijayvargiya G. DBNotes: A post-it system for relational databases based on provenance [C] // Proc of the 24th ACM SIGMOD-SIGACT-SIGART Symp on Principles of Database Systems. New York: ACM, 2005: 942-944

[12] Widom J Trio. A system for integrated management of data, accuracy, and lineage [C] //Proc of the 2nd Biennial Conf on Innovative Data Systems Research. New York: ACM, 2005: 262-276

[13] Karvounarakis G, Ives I G, Tannen V. Querying data provenance [C] //Proc of the ACM SIGMOD Int Conf on Management of Data. New York: ACM, 2010: 951-962

[14] Glavic B, Alonso G Perm. Processing provenance and data on the same data model through query rewriting[C]//Proc of the 25th IEEE Int Conf on Data Engineering. Piscataway, NJ: IEEE, 2009: 174-185

[15] Green T J. Containment of conjunctive queries on annotated relations [J]. Theory of Computing System, 2011, 49(2): 429-459

[16] Green T J, Huang S S, Loo B T, et al. Datalog and recursive query [J]. Processing Foundations and Trends in Databases, 2013, 5(2): 105-195

[17] Deutch D, Milo T, Roy S, et al. Circuits for datalog provenance [C] //Proc of the 17th Int Conf on Data Theory. New York: ACM, 2014: 201-212

Xue Jianxin, born in 1984. PhD candidate. Student member of China Computer Federation. His research interests include recursive query evaluation and data provenance (xuejianxin @ research. neu. edu. cn).

Shen Derong, born in 1964. Professor and PhD supervisor of the Northeastern University. Senior member of China Computer Federation. Her main research interests include distributed database, Web data management and Web services.

Kou Yue, born in 1980. Associate professor at Northeastern University, China. Member of China Computer Federation. Her main research interests include Web search, Web mining, and dataspace.

Nie Tiezheng, born in 1980. Associate professor at Northeastern University, China. Member of China Computer Federation. His main research interests include data quality and data integration.

Yu Ge, born in 1962. Professor and PhD supervisor at Northeastern University, China. His main research interests include database theory and technology, distributed and parallel systems, and network information security.