软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2012,23(3):550−564 [doi: 10.3724/SP.J.1001.2012.04050]
©中国科学院软件研究所版权所有.
E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

n-of-*N* 数据流模型上高效概率 Skyline 计算*

杨永滔+, 王意洁

(国防科学技术大学 计算机学院 并行与分布处理国家重点实验室,湖南 长沙 410073)

Efficient Probabilistic Skyline Computation Against *n*-of-*N* Data Stream Model

YANG Yong-Tao+, WANG Yi-Jie

(National Key Laboratory for Parallel and Distributed Processing, College of Computer, National University of Defense Technology, Changsha 410073, China)

+ Corresponding author: E-mail: ytyang@nudt.edu.cn

Yang YT, Wang YJ. Efficient probabilistic skyline computation against *n*-of-*N* data stream model. *Journal of Software*, 2012,23(3):550−564. http://www.jos.org.cn/1000-9825/4050.htm

Abstract: This paper studies the problem of computing *q*-skylines against probabilistic data streams. Compared with the existing methods, which only support the sliding window model, this method can support the more general *n*-of-*N* data stream model. This method of transforming *q*-skyline queries is used for the stabbing queries on an interval tree to support *n*-of-*N* model. The paper proposes an algorithm, named PnNM, to maintain the data structures, which is needed for supporting *n*-of-*N* model. The PnNM algorithm can efficiently handle the update of the candidate set of uncertain data objects and the updates of the intervals. An algorithm, named PnNCont, is also proposed to handle continuous *q*-skyline queries against *n*-of-*N* model. The theoretical analyses and extensive experiments demonstrate that this algorithms can be very efficient in handing *q*-skyline queries against probabilistic data streams under *n*-of-*N* model.

Key words: probabilistic data stream; skyline; *n*-of-*N* model; sliding window; continuous query

摘 要: 研究概率数据流上的 *q*-skyline 计算问题.与只支持滑动窗口数据流模型的已有方法相比,所提出的方法能够支持更为通用的 *n*-of-*N* 数据流模型.采用将 *q*-skyline 查询转换为区间树上刺入查询的方法支持 *n*-of-*N* 数据流模型.提出 PnNM 算法维护支持 *n*-of-*N* 数据流模型所需的相关数据结构,高效处理了不确定对象候选集合更新和区间更新等维护工作;提出 PnNCont 算法实现连续查询处理.理论分析和实验结果表明,算法能够有效地支持概率数据流 *n*-of-*N* 模型上的 *q*-skyline 查询处理.

关键词: 概率数据流;skyline;*n*-of-*N* 模型;滑动窗口;连续查询

中图法分类号: TP311 **文献标识码**: A

Skyline 计算在多目标决策中起着十分重要的作用,自 Börzsönyi 等人[1]首次将 skyline 查询引入数据库领域以来,skyline 得到了广泛的研究.Skyline 研究不但针对 Börzsönyi 等人针对的静态、确定的常规数据集有很多

* 基金项目: 国家自然科学基金(60873215); 国家重点基础研究发展计划(973)(2011CB302601); 湖南省自然科学杰出青年基金(S2010J5050); 高等学校博士学科点专项科研基金(200899980003)

收稿时间: 2010-02-10; 修改时间: 2010-08-13; 定稿时间: 2011-04-20

© 中国科学院软件研究所 http://www.jos.org.cn

高效的算法[2−4],而且还扩展到偏序域数据集[5]、分布数据集[6,7]、动态数据集[8]以及不确定数据集[9]上.

数据流是一种动态更新的数据集,处理数据流上查询常用的模型是滑动窗口模型.令滑动窗口长度为 N,则滑动窗口模型实际上是对数据流中最近 N 个对象进行处理.*n*-of-*N* 数据流模型[10]扩展了滑动窗口模型,能够针对数据流上最近任意 $n(n\leqslant N)$个对象进行处理.Lin 等人[11]首先研究了 *n*-of-*N* 数据流模型上的 skyline 计算,但 Lin 的工作仅针对确定数据流.概率数据流是动态更新且不确定的数据集,Zhang 等人[12]首先研究了概率数据流上概率 skyline 的计算问题,提出连续计算滑动窗口中 N 个对象的 q-skyline[9](q 为概率阈值)的方法,但 Zhang 的方法支持更为通用的 *n*-of-*N* 数据流模型上.

为支持 *n*-of-*N* 数据流模型上的 q-skyline 查询,首先要解决的问题是如何采用一种高效的机制将仅针对整个滑动窗口的 q-skyline 计算扩展到能够针对滑动窗口中任意最近 n 个对象进行计算.为此,本文采用将 q-skyline 查询转换为区间树上刺入查询的方法.本文为支持概率数据流 *n*-of-*N* 模型上的 q-skyline 查询所做工作如下:

1) 提出了基于 R 树的 RDO 树索引数据对象,在确定候选集合中被最新到达对象支配的对象时减小了搜索空间;
2) 提出 PnNM 算法以维护 RDO 树、区间树等数据结构,高效处理了不确定对象候选集合更新和区间更新等维护工作;
3) 提出 PnNCont 算法处理 *n*-of-*N* 模型上的连续 q-skyline 查询.

1 背景知识

1.1 相关工作

自从 Börzsönyi 等人[1]将 skyline 研究引入数据库研究领域后,skyline 计算迅速成为研究热点之一.据本文作者所知,迄今尚无其他支持 *n*-of-*N* 模型上的概率 skyline 计算的研究工作.与本文工作研究工作比较密切的是 Lin[11],Pei[9]和 Zhang 等人[12]关于数据流上、不确定数据上以及概率数据流上 skyline 计算的研究.Pei 等人[9]首先研究了不确定数据集上进行 skyline 分析的问题,基于可能世界(possible world)语义模型定义了不确定数据集上 skyline,即 p-skyline(p 表示概率阈值),并提出自顶向下和自底向上两个算法.Pei 所用的不确定数据模型是用多维空间中的数个点描述一个不确定对象,每个点可以当作不确定对象的一个实例.与此不同,本文的不确定数据模型是用多维空间的一个点以及相应的出现概率表示一个不确定对象,更适合描述概率数据流应用[12].Lin 等人[11]首先研究了 *n*-of-*N* 数据流模型[10]上的 skyline 计算.他们首先表明,支持该查询所需保存的数据对象数量是对数量级的,因此适合数据流环境下的处理;他们还巧妙通过编码将 skyline 计算转换为区间树上的刺入查询(stabbing queries),但是他们的算法处理的是确定数据流,不能用于不确定数据流.Zhang 等人[12]研究的是概率流上滑动窗口的 skyline 计算.Zhang 等人证明,为处理概率流上 skyline 计算而需要在内存中保存的对象数目是对数量级的,并且提出基于 R 树的计算方法.然而,他们的算法只能用于计算整个滑动窗口上的概率 skyline,不能处理更为通用的 *n*-of-*N* 模型,而这正是本文所要研究的.

国内的研究者也对 skyline 计算进行了深入的研究,如数据流上的 skyline 查询[13−15]等工作.孙圣力等人[15]研究了概率数据流上 skyline 查询处理,提出基于网格索引技术并运用概率定界等启发式规则的 SOPDS 算法,但其方法也不支持 *n*-of-*N* 模型.

1.2 概念和定义

本文所讨论的数据对象(简称对象)是 d-维空间的一个点,不确定对象 e 是具有出现概率 $P(e)(0<P(e)\leqslant 1)$的对象($P(e)$表示 e 出现或存在的概率).本文讨论的概率数据流(简称概率流)由一系列不确定对象组成,记作 DS;对象按序到达,$\kappa(e)$是对象 e 的时间标签(time label),表示对象 e 是第$\kappa(e)$个到达.DS 是 append-only 的,DS_N 表示包含当前最近 N 个元素的滑动窗口.此外,本文研究针对的是低维空间.

定义 1(支配关系). 给定 d-维空间的两个对象 a 和 b,它们在第 i 维的属性值分别记作 $a.i$ 和 $b.i$.我们可以定

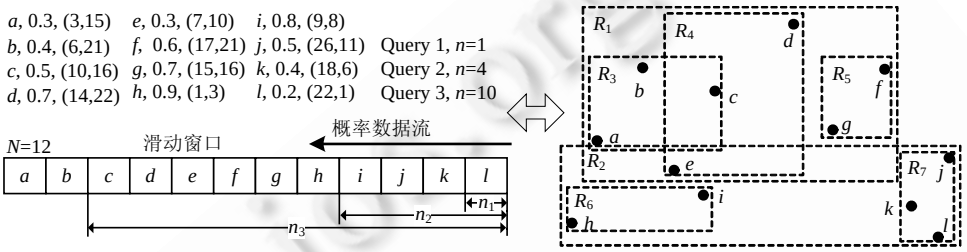
© 中国科学院软件研究所 http://www.jos.org.cn

义它们之间的支配关系如下:a 支配 b(记作 $a \prec b$)当且仅当对于所有 $i(1\leqslant i\leqslant d)$均有 $a.i\leqslant b.i$,并存在 $i_0(1\leqslant i_0\leqslant d)$使得 $a.i_0<b.i_0$.N 个对象的 skyline 就是所有那些不被其他对象支配的对象构成的子集.

定义 2(n-of-N 数据流模型). N 表示包含最近 N 个对象的当前滑动窗口,$n(n\leqslant N)$表示最近的 n 个对象(包含在有 N 个对象的滑动窗口之内).n-of-N 模型要能够在任意 $n(n\leqslant N)$个最近对象上进行查询操作,大小为 N 的滑动窗口只是 $n=N$ 时的一个特例.本文用 DS_n 表示最近 $n(n\leqslant N)$个对象.

n-of-N 模型的意义在于 N 反映所能获得的系统资源能够处理的最长窗口,而 n 反映了用户对数据时效性的要求.比如,股票投资者在购买股票之前,想通过已经执行的交易信息(交易量和交易价格)的 skyline 进一步了解市场走势[11],而且往往希望可以同时获得过去一天、一周或一月等不同时间范围内的信息.n-of-N 模型对此提供了有效支持,只需设定 n 为一天、一周或一月等时间范围内的数据对象数目并再查询一次,就可以得到相应结果.

图 1 给出二维情况下概率流滑动窗口示例,其中,对象的格式是"对象名称、出现概率、对象坐标",概率数据流从滑动窗口的右端向左端流动.



(a) 概率流滑动窗口 (b) 滑动窗口中对象在二维平面上的分布示意

Fig.1 An example of the sliding window of a probabilistic data stream

图 1 概率流滑动窗口示例

定义 3(非冗余集合 R_N[11]). 对象 e 如果被比其晚到达的对象支配或者过期(不再属于 DS_N),则对象 e 称为冗余的.R_N 即是 DS_N 中所有非冗余对象的集合,即 $\forall e\in R_N$,不存在 $a\in DS_N$,使得 $\kappa(a)>\kappa(e)$且 $a\prec e$.

DS_N 的可能世界(possible world)定义为 DS_N 的一个子序列[12],本文也把可能世界称为实例.可能世界 W 出现的概率记作 $P(W)$,

$$P(W)=\prod_{e\in W}P(e)\times\prod_{e\notin W}(1-P(e)),$$

其中,$e\in DS_N$.令 Ω 是所有可能世界的集合,则有 $\sum_{W\in\Omega}P(W)=1$.令 $SKY(W)$表示 W 的 skyline,则存在一个可能世界,且其 skyline 包括对象 e 的概率称为 e 的 skyline 概率,记作 $P_{sky}(e)$,且有 $P_{sky}(e)=\sum_{e\in SKY(W),W\in\Omega}P(W)$.根据可能世界的性质,我们还可以定义对象的 skyline 概率如下:

定义 4(对象的 skyline 概率[12]). $P_{sky}(e)=P(e)\times\prod_{a\in DS_N,a\prec e}(1-P(a))$.

定义 5(q-skyline[12]). 给定概率阈值 $q(0<q\leqslant 1)$,可以定义 q-skyline 为 DS_N 中所有 skyline 概率大于等于 q 的对象的集合,记作 $SKY_{N,q}$,即 $SKY_{N,q}=\{e|e\in DS_N\wedge P_{sky}(e)\geqslant q\}$.相应地,$DS_n$ 上的 q-skyline 记作 $SKY_{n,q}$.

定义 5 中的 q-skyline 即是本文讨论的概率 skyline.本文研究的问题是,给定概率阈值 q,对于任意 $n(n\leqslant N)$,在概率流上连续地计算 $SKY_{n,q}$.本文也称 $SKY_{n,q}$ 的计算为 q-skyline 查询.

Zhang 等人[12]根据定义 4 中 $P_{sky}(e)$计算公式中支配 e 的对象是早于 e 还是晚于 e 到达,定义了

$$P_{new}(e)=\prod_{a\in DS_N,a\prec e,\kappa(a)>\kappa(e)}(1-P(a)),\ P_{old}(e)=\prod_{a\in DS_N,a\prec e,\kappa(a)<\kappa(e)}(1-P(a)),$$

并在此基础上定义并证明了计算 $SKY_{N,q}$ 所必须在内存中保存的数据对象集合,如下所示:

定义 6(候选集合 $S_{N,q}$[12]). $S_{N,q}=\{e|e\in DS_N\wedge P_{new}(e)\geqslant q\}$.

© 中国科学院软件研究所 http://www.jos.org.cn

Zhang 等人[12]证明,在 DS_N 上计算 $SKY_{N,q}$,与在 $S_{N,q}$ 上计算 $SKY_{N,q}$ 是等价的.因此,只需以 $S_{N,q}$ 为候选集合,而无需考虑整个滑动窗口.类似地,在 DS_n 上可定义 $S_{n,q}$.

Lin 等人[11]针对的是确定数据流,首先提出将 skyline 查询转化为区间树上刺入查询的思想.Lin 方法中的两个关键概念是在 R_N 上定义的关键支配关系(critical dominance relation)和后向关键支配关系(backward critical dominance relation).本文在 $S_{N,q}$ 上定义了类似的关系,并说明了这两个概念在 $S_{N,q}$ 上也是良定义的.

定义 7(对象 e 的关键支配关系). 在 $S_{N,q}$ 中,如果 e'是早于 e 到达并支配 e 的到达最晚的对象,那么 e'支配 e 就称作对象 e 的关键支配关系.e'称为 e 的关键祖先(critical ancestor),记作 a_e.如果 a_e 存在,那么$\kappa(a_e)$满足关系:$\kappa(a_e)=\max\{\kappa(e')|e'\prec e\wedge\kappa(e')<\kappa(e)\}$;如果 a_e 不存在,那么本文假定$\kappa(a_e)$为 0.

定义 8(对象 e 的后向关键支配关系). 在 $S_{N,q}$ 中,如果 e'是晚于 e 到达并支配 e 的到达最早的对象,那么 e'支配 e 就称作对象 e 的后向关键支配关系.e'称为 e 的后向关键祖先(backward critical ancestor),记作 b_e.如果 b_e 存在,那么$\kappa(b_e)$满足关系:$\kappa(b_e)=\min\{\kappa(e')|e'\prec e\wedge\kappa(e')>\kappa(e)\}$.

定义 7 和定义 8 都是良定义的.文献[11]在 R_N 上定义的关键支配关系和后向关键支配关系满足一个性质:如果 a_e 和 b_e 不是因为过期而从 R_N 中移除的话,那么 e 也必然已从 R_N 中移除.定义 7 和定义 8 中定义的 a_e 和 b_e 也满足类似性质(对 $S_{N,q}$ 而言),见定理 1.

定理 1. 如果 a_e 和 b_e 不是因为过期而从 $S_{N,q}$ 中移除的话,那么 e 肯定已经从 $S_{N,q}$ 中移除.

证明:a_e 不是因为过期而从 $S_{N,q}$ 中移除,那么 a_e 是因为 $P_{new}(a_e)<q$ 而从 $S_{N,q}$ 中移除.对于对象 c 而言,如果$(1-P(c))$是用于计算 $P_{new}(a_e)$的因子,则有 $c\prec a_e$ 且$\kappa(c)>\kappa(a_e)$,则必有 $c\prec e$,并且$\kappa(c)>\kappa(e)$;不然的话,$\kappa(c)$满足$\kappa(a_e)<\kappa(c)<\kappa(e)$,则与定义 7 中 a_e 的定义矛盾.因此,$(1-P(c))$也是用于计算 $P_{new}(e)$的因子,则 $P_{new}(e)\leqslant P_{new}(a_e)<q$,对象 e 必须从 $S_{N,q}$ 中删除.

根据定义 8,b_e 晚于 e 到达,则晚于 b_e 到达并支配 b_e 的对象也晚于 e 到达并支配 e,因此对于对象 c,如果$(1-P(c))$是 $P_{new}(b_e)$中因子,那么也必定是 $P_{new}(e)$中的因子,因此 $P_{new}(e)\leqslant P_{new}(b_e)$.因此,如果 b_e 因为 $P_{new}(b_e)<q$ 而从 $S_{N,q}$ 中删除,那么 e 也会因为 $P_{new}(e)<q$ 而从 $S_{N,q}$ 中删除. □

1.3 查询处理流程概述

算法 0 描述了本文提出的 *n*-of-*N* 模型下 skyline 查询处理系统,其主要处理流程是:新数据对象的到达激活维护模块(PnNM 算法)进行更新维护操作,维护完成后,新对象便加入到窗口中;然后进入查询执行模块,处理用户提交的即时或连续查询,返回用户结果集后,再次等待新数据达到,进入下一个处理周期.

算法 0. 查询系统处理流程.

描述:

1. **while** 新对象到达 **do**
2. PnNM 维护算法;
3. 查询处理;
4. **end while**

2 维护候选集合和区间树的 PnNM 算法

2.1 存储和索引结构

计算 *n*-of-*N* 模型上的 *q*-skyline 需要维护候选数据对象集合为 $S_{N,q}$,为此,本文首先提出基于 R 树的双层索引结构 RDO 树存储 $S_{N,q}$ 中的对象;对于对象的区间和时间标签这两个重要属性,分别采用区间树和扩展后的红黑树(称为时间标签树)来存储.对于任意 $e\in S_{N,q}$,其区间记作 $I(e)$、时间标签记作$\kappa(e)$,则 RDO 树、区间树和时间标签树分别存储 e,$I(e)$和$\kappa(e)$,并且它们之间可以通过指针相互访问.如图 2 所示.

© 中国科学院软件研究所 http://www.jos.org.cn

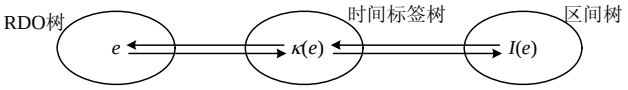


Fig.2 Maps among the three trees

图 2 3 种树结构之间的映射关系

2.1.1 RDO 树

定理 2. 非冗余集 R_N 是 $S_{N,q}(0<q\leqslant 1)$的子集.

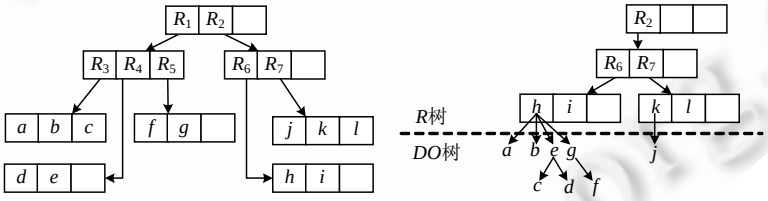
证明:任给 $e\in R_N$,e 不被 DS_N 中任何晚于 e 到达的对象支配,则 $P_{new}(e)=1$,因此对于任意 $q(0<q\leqslant 1)$,e 都属于 $S_{N,q}$. □

根据定理 2,候选集合 $S_{N,q}$ 可以分为 R_N 和$(S_{N,q}-R_N)$两个部分.本文在内存中建立 R 树[16]索引 R_N 中的对象,至于$(S_{N,q}-R_N)$中的对象,都作为某个对象的 DO 集中的元素出现.对象的 DO 集定义如下:

定义 9(对象 e 的 DO 集). $S_{N,q}$ 中所有后向关键支配祖先为 e 的对象集合称为对象 e 的 DO 集(名字取 dominated by e 和 older than e 的首字母缩写),记作 $DO(e)$.

若 $a\in DO(e)$则 $\kappa(a)<\kappa(e)$且 $P_{new}(a)<P_{new}(e)$.当 e 到达时, 原来 R_N 中被 e 支配并且更新(乘以$(1-P(e))$)后的 P_{new} 值仍然大于等于 q 的对象组成最初的 $DO(e)$.$DO(e)$在 e 到达时确定后,只会因为其中对象从 $S_{N,q}$ 中被删除而缩小.此外,$DO(e)$中的对象 a 也可能有 $DO(a)$,依此类推;这个层次结构组成了以 e 为根的一个多叉树,我们称 e 的 DO 树,记作 $DOTree(e)$.对象 e 相应的最小包围矩形(MBR)就不再仅仅包含 e 这个点,而是整个 $DOTree(e)$(包括根 e 在内)的包围矩形.

任给 $e\in S_{N,q}$,若 $e\in R_N$,则 e 没有后向关键支配祖先,$DOTree(e)$不是其他某个对象 a 的 $DOTree(a)$的子树;若 $e\notin R_N$,则 e 必然有后向关键支配祖先,并在 R_N 中存在唯一的对象 e'作为其后向关键支配关系的远祖,即 $DOTree(e)$必是 $DOTree(e')$的子树.如此则可用双层树结构存储 $S_{N,q}$:上层为 R 树,存储 R_N;下层由 R_N 中每个对象的 DO 树构成,存储$(S_{N,q}-R_N)$.我们称该双层树结构为 RDO 树(R 树和 DO 树的合记).图 3 给出了存储图 1 所示对象的 RDO 树的基本结构,并与仅使用 R 树的情形对比.为简明起见,图 3 中取概率阈值 q=0.001,以使得滑动窗口中的所有对象均保留在 $S_{N,q}$ 中.



(a) 仅使用 R 树的存储示意图 (b) 使用 RDO 树的存储示意图

Fig.3 Structure of a RDO tree

图 3 RDO 树的结构示意图

本文也将 RDO 树记为 $R(S_{N,q})$.采用 RDO 树的好处是,当对象 e 被新到达的对象 e_{new} 支配时,$DOTree(e)$中的其他对象也肯定被 e_{new} 支配,因而减少了寻找 $S_{N,q}$ 中所有被 e_{new} 支配的对象的开销.

2.1.2 时间标签树

每个对象都有一个时间标签,表示对象的相对到达顺序.时间标签树基于红黑树[17],存储 $S_{N,q}$ 中所有对象的时间标签.红黑树每个结点存储一个对象的时间标签(因为时间标签与对象一一对应,因此逻辑上可认为同时存储了相应对象),并且扩展每个子结点使其同时存储以其为根的子树中的所有对象的最小包围矩形.增加时间标签树的目的是提高区间树中区间更新的效率,本文将时间标签树记作 $L(S_{N,q})$.

© 中国科学院软件研究所 http://www.jos.org.cn

2.1.3 区间树和即时查询算法

Lin等人[11]将对象e映射到区间$(\kappa(a_e),\kappa(e)]$,从而将DS_n上的skyline查询转化为区间树上刺入查询[18].令M表示当前到达的总对象个数,则根据Lin 的方法,如果$M-n+1\in(\kappa(a_e),\kappa(e)]$,那么$e$是$DS_n$的skyline中对象.此即意味着,$(\kappa(a_e),\kappa(e)]$表示的是使得$e$属于$DS_n$的skyline的$M-n+1$的有效范围.根据这一分析,本文重新定义了区间左端点(右端点仍为$\kappa(e)$,不再变化),以使得q-skyline查询也能够转换为区间树上的刺入查询.

定义10(e的区间左端点). 假定$S_{N,q}$中早于e到达并支配e的对象有m个,并且按照到达顺序从早到晚分别记作$c_1,\ldots,c_m$,令$l=\min\{k|1\leqslant k\leqslant m\wedge P(e)P_{new}(e)\prod_{i=k}^{m}(1-P(c_i))\geqslant q\}$,其中,$P(e)P_{new}(e)\prod_{i=k}^{m}(1-P(c_i))$也就是在$[\kappa(c_k),\kappa(c_m)]$范围内计算的$P_{sky}(e)$.如果$l$存在,那么$e$的区间左端点就为$\kappa(c_{l-1})(l>1)$或者$0(l=1)$,并且用$e$的属性值$skypob_old$记录$P(e)P_{new}(e)\prod_{i=l}^{m}(1-P(c_i))$(其他情况下,$skypob_old$设为0).如果不存在这样的$l$,则分为两种情况:如果$P(e)P_{new}(e)<q$,那么无需在区间树中插入对象$e$相关的区间;如果$P(e)P_{new}(e)\geqslant q$,那么$m=0$时左端点为$0,m\neq 0$时左端点为$\kappa(c_m)$(即$\kappa(a_e)$).显然,按照上述方法定义的对象$e$的区间$(k_0,\kappa(e))$同样具有如下性质:如果$M-n+1\in(k_0,\kappa(e))$,则$e$是$DS_n$的$q$-skyline中的对象.本文用动态区间树(记为$I(S_{N,q})$)索引$S_{N,q}$中对象按上述方法定义的所有区间,我们可立即得到定理3.

定理3. 以$M-n+1$为刺入点在$I(S_{N,q})$上执行刺入查询,输出区间右端点对应的对象组成的集合就是$SKY_{n,q}$.

易知定理3成立.因为如果$(l_1,l_2]$是刺入查询输出的区间之一,则按照定义10,其右端点l_2对应的对象属于$SKY_{n,q}$.根据定理3,即时查询就转化为是在区间树$I(S_{N,q})$上以$M-n+1$为刺入点的刺入查询操作.

2.2 PnNM算法

2.2.1 算法框架

算法1描述了本文提出的维护$S_{N,q}$和$I(S_{N,q})$的PnNM算法.最新对象e_{new}到达时,PnNM算法首先检查$S_{N,q}$中是否有过期的对象要删除.如果e_{old}是过期的对象,那么直观上e_{old}的过期会影响$S_{N,q}$中那些被e_{old}支配的对象,但定理4表明,实际上不作处理也不会影响查询结果.

定理4. 最新对象e_{new}到达时,e是不被e_{new}支配的对象,其对应的区间为$(l,\kappa(e)]$.令过期的时间标签为$M-N+1(M$为e_{new}到达之前已到达的对象总数),则e不更新区间左端点也不影响查询结果的正确性.

证明:根据定义10,只有$M-N+1=l$时,对象e的区间左端点才需要更新成0.也就是说,e的实际区间应该为$(0,\kappa(e)]$.如果不更新,则e的区间保持为$(M-N+1,\kappa(e)]$.最新对象e_{new}到达后,对于任意$n\leqslant N$,区间树$I(S_{N,q})$上执行刺入查询的刺入点为$((M+1)-n+1)$.显然$(M+1)-n+1>M-N+1$,则$((M+1)-n+1)$属于区间$(0,\kappa(e)]$等价于$((M+1)-n+1)$属于$(M-N+1,\kappa(e))$,故定理成立. □

算法1. PnNM算法:维护$S_{N,q}$以及$I(S_{N,q})$.

描述:

1. **while** 最新对象e_{new}到达 **do**
2. 检查并处理过期的对象;
3. 确定被e_{new}支配的$A(e_{new})\subseteq S_{N,q}$;
4. $A(e_{new})$的预处理; /*算法2*/
5. 将C中对象按后向关键祖先的时间标签从大到小排序;
6. 更新$DO(e_{new})\cup NR(e_{new})$中对象的区间; /*算法4*/
7. 将C中的对象从$S_{N,q}$中删除;
8. 确定e_{new}的关键支配祖先及其MBR;
9. 将$\kappa(e_{new})$,e_{new}及其区间分别插入到$L(S_{N,q})$,$R(S_{N,q})$和$I(S_{N,q})$中;
10. **end while**

算法1中,Line 3中的$A(e_{new})$并没有包括$S_{N,q}$中所有被e_{new}支配的对象,而只是各DO树中被e_{new}支配的子树的根结点,也即当$a\in A(e_{new})$时,如果$b\in DOTree(a)$,那么必有$b\notin A(e_{new})$.本文利用DO树避免查找$S_{N,q}$中所有被

© 中国科学院软件研究所 http://www.jos.org.cn

e_{new}支配的对象.至于确定 $A(e_{new})$的办法,本文先采用深度优先搜索[11]搜索 RDO 树:搜索 R 树时用深度优先搜索,如果需要搜索 R 树中叶结点中保存的某个对象的 DO 树,采取的也是深度优先的搜索方法.搜索过程中,根据 e_{new}与结点的 MBR 的支配关系[11]决定是否扩展某个结点进行进一步的搜索.

Line 4 是将 $A(e_{new})$划分为 C,$DO(e_{new})$和 $NR(e_{new})$这 3 个子集的预处理过程,其中:$DO(e_{new})$和 $NR(e_{new})$中对象的区间在之后的处理过程中被更新;C 暂时保存本次更新需从 $S_{N,q}$ 中删除的对象,用于优化区间的更新.具体预处理过程见第 2.2.2 节.

Line 5 是为了使用 C 中的对象而对 C 作预处理.Line 6 是更新区间的过程(参见第 2.2.3 节).Line 7 将 C 中的对象从 $S_{N,q}$ 中删除.本文采用批量删除方法实现该删除过程,即不是每删除一个对象就调整 RDO 树的 R 树部分,而是删除所有对象后再调整 R 树.Line 8 和 Line 9 是将 e_{new} 加入到算法维护的数据结构之中,其中,Line 8 确定 e_{new}的关键支配祖先采取的搜索方法与确定 $A(e_{new})$的方法是类似的,不同的是前者搜索支配 e_{new}的对象,后者搜索被 e_{new} 支配的对象.

2.2.2 $A(e_{new})$的预处理

$A(e_{new})$的预处理是将 $A(e_{new})$划分为 C,$DO(e_{new})$和 $NR(e_{new})$这 3 个子集的过程.划分算法首先以$(1-P(e_{new}))$更新对象的 P_{new},$P_{new,\min}$ 和 $P_{new,global}$ 等值.如果更新后 $P_{new}<q$,那么该对象加入集合 C;如果更新后 $P_{new}\geqslant q$,那么原本属于 R_N 的对象加入集合 $DO(e_{new})$,原本不属于 R_N 的对象就加入集合 $NR(e_{new})$.

算法 2 描述了 $A(e_{new})$的具体划分过程.在划分过程中,如果对象 e 属于 R_N,那么可以同时确定 b_e 为 e_{new}.此外,对于当前集合 C 中的对象 a,因为 $P_{new}(a)<q$,因此 $DOTree(a)$中除 a 之外的其他对象也有 $P_{new}<q$,并且这些对象除了需要从 $S_{N,q}$ 中删除外,无需进一步处理.也就是说,$DOTree(a)$树中处理根结点 a 即可,因此节省了部分开销.

算法 2. $A(e_{new})$的预处理.

1. **for** $\forall e\in A(e_{new})$ **do**
2. 以$(1-P(e_{new}))$更新 $P_{new}(e)$,$P_{new,\min}(e)$和 $P_{new,global}(e)$;
3. **if** $P_{new}(e)<q$ **then**
4. **if** $e\in R_N$ **then** $b_e\leftarrow e_{new}$; **end if**
5. e 加入 C;
6. 删除 $DOTree(e)$中的其他对象;
7. **else**
8. **if** $e\in R_N$ **then**
9. $b_e\leftarrow e_{new}$;e 加入 $DO(e_{new})$;
10. **else** e 加入 $NR(e_{new})$; **end if**
11. 更新 $DOTree(e)$中其他对象的 P_{new} 值; /*算法 3*/
12. **end if**
13. **end for**

对于 $DO(e_{new})$和 $NR(e_{new})$中的对象 e,虽然更新后 $P_{new}(e)\geqslant q$,但是其 DO 树中仍然可能会有 $P_{new}<q$ 的对象,因此需要进一步检查(算法 2 中的 Line 11).

算法 2 结束后,集合 C 最终包含了 $S_{N,q}$ 中被 e_{new} 支配、并且更新后的 P_{new} 小于 q 以及后向关键祖先不被 e_{new} 支配的对象.也就是说,若 $a\in C$,则 $e_{new}\prec a$,更新后 $P_{new}(a)<q$,并且如果对象 $b\in DOTree(a)$且 $b\neq a$,那么 $b\notin C$. $DO(e_{new})$和 $NR(e_{new})$中对象都满足 $P_{new}\geqslant q$,$DO(e_{new})$包扩了 $S_{N,q}$ 中所有被 e_{new} 支配并且在 e_{new} 到达前原属于 R_N 的对象,$NR(e_{new})$包括了所有 $S_{N,q}$ 中所有被 e_{new} 支配并且在 e_{new} 到达前不属于 R_N 的对象.

算法 3 描述了算法 2 中 Line 11 更新 $DOTree(e)$中其他对象的 P_{new} 值的过程.对于 $e\in DO(e_{new})\cup NR(e_{new})$,为了提高更新 $DOTree(e)$中对象的 P_{new} 值的效率,本文用 $P_{new,\min}(e)$表示 $DOTree(e)$中所有对象中最小的 P_{new},$P_{new,global}(e)$表示 $DOTree(e)$中计算 P_{new} 所需要乘上的公共因子(初始值为 1).当更新后 $P_{new,\min}(e)\geqslant q$,那么 $DOTree(e)$就可以暂时不更新.如果 $P_{new,\min}(e)<q$,那么 $DOTree(e)$中必定有 $P_{new}<q$ 的结点,因此需要进一步更新

© 中国科学院软件研究所 http://www.jos.org.cn

DOTree(*e*)中其他对象的 P_{new} 值,并将 $P_{new}<q$ 的对象加入到 *C* 中.

算法 3. 更新 *DOTree*(*e*)中其他对象的 P_{new}(*update_Pnew*(*e*)).

描述:

```
1.  if P_new(e)≥q then
2.    if P_new,min(e)<q then
3.      for e′∈DO(e) do
4.        P_new(e′)←P_new(e′)P_new,global(e);
5.        if P_new(e′)<q then
6.          将 e′加入集合 C;
7.          删除 DOTree(e′)中的其他对象;
8.        else
9.          P_new,global(e′)←P_new,global(e′)P_new,global(e);
10.         P_new,min(e′)←P_new,min(e′)P_new,global(e);
11.         将 e′加入 NR(e_new);
12.         update_Pnew(e′);
13.       end if
14.     end for
15.     P_new,global(e)←1;
16.   end if
17. end if
```

2.2.3 区间的更新过程

算法 4 描述了 PnNM 算法框架中 $DO(e_{new})\cup NR(e_{new})$中对象的区间的更新过程(算法 1 中的 Line 6),实际上需要更新的只是区间的左端点(对象 *e* 相应区间的左端点用 *e.lower* 表示).算法 4 调用 *update_lower_end*()(算法 5)分别处理了 $DO(e_{new})$和 $NR(e_{new})$中对象的区间更新.

算法 4. 更新 $DO(e_{new})\cup NR(e_{new})$中对象的区间.

描述:

```
1. update_lower_end(e_new);                                  /*算法 5*/
2. 构建临时对象 e_tmp,使得 P_new,global(e_tmp)=1,并且所有 NR(e_new)中的对象都属于 DO(e_tmp).B;
3. update_lower_end(e_tmp);
```

算法 5 中描述了具体的区间更新过程,即算法 4 中调用的 *update_lower_end*()操作.区间 *I*(*e*)更新的基本方法是,在原先的区间中搜索支配 *e* 的对象,以计算新的区间左端点.因为区间表示一个时间标签范围,所以本文在对象的时间标签上建立一个基于红黑树的索引($L(S_{N,q})$),并为红黑树的结点增加最小包围矩形(MBR)——包含了以该结点为根的子树中所有对象结点,然后在此红黑树上在某个区间范围内进行搜索.当然,为了提高搜索效率,本文也提出了一些启发式规则,决定是否需要进行更新以及缩小搜索的区间范围.

算法 5. 更新 *DOTree*(*a*)中相关对象的区间左端点(*update_lower_end*(*a*)).

描述:

```
1.  for ∀e∈DO(a) do
2.    P_new(e)←P_new(e)P_new,global(a);
3.    P_new,min(e)←P_new,min(e)P_new,global(a);
4.    P_new,global(e)←P_new,global(e)P_new,global(a);
5.    if P(e)P_new(e)<q then
6.      将 e 的区间从 I(S_N,q)删除;
```

© 中国科学院软件研究所 http://www.jos.org.cn

```
7.    else
8.      e.skyprob_old←e.skyprob_old*(1−P(e_new));
9.      if e.skyprob_old<q then
10.       if κ(a_e)>M−N+1 then
11.         if (1−P(a_e))P(e)P_new(e)<q then
12.           将 e 的区间左端点更新为κ(a_e);
13.         else
14.           if e∈DO(a).A then
15.             l←max{a.lower,e.lower};
16.           else l←e.lower; end if
17.           k_e←find_dominator_in_C(e,l);
18.           update_exact_lower_end(e,max{k_e,l});
19.         end if
20.       end if
21.     end if
22.   end if
23.   P_new,global(a)←1;
24.   update_lower_end(e);
25  end for
```

算法 5 的输入参数为对象 a,并以深度优先的模式处理 $DOTree(a)$中各对象的区间的更新.对于当前处理的对象 e,先通过一些规则判定是否需要更新区间左端点(算法 5 中的 Line 9 利用 $e.skyprob\_old$ 进行判断).如果确定需要更新(算法 5 中的 Line 5 和 Line 6),则先确定搜索区间,再调用 $update_exact_lower_end()$确定最新的区间左端点.

2.2.4 区间更新过程的优化

本文在算法 5 描述的区间更新过程中应用了 4 条启发式优化规则.

规则 1. 对象 e_{new} 到达后,如果 $e_{new}\prec e$,并且 $P(e)P_{new}(e)<q$,那么 e 相应的区间可以从 $I(S_{N,q})$中删除.

规则 2. 对象 e_{new} 到达后,若 $e_{new}\prec e$,a_e 存在,并且$(1-P(a_e))P(e)P_{new}(e)<q$,则 e 的区间的左端点应更新为$\kappa(a_e)$.

规则 3. 将 $DO(a)$分为 A,B 两个部分:对于 $e\in DO(a)$,如果 $P(e)(1-P(a))\leqslant P(a)$,那么 e 属于 A 部分(记为 $DO(a).A$);否则,e 属于 B 部分(记为 $DO(a).B$).如果 $e\in DO(a).A$,则 $P_{sky}(e)\leqslant P_{sky}(a)$总是成立的,因此 $e.lower\geqslant a.lower$ 也总是成立的.

规则 1(算法 5 中 Line 5~Line 6)避免更新那些将被删除的区间.根据区间左端点的定义(定义 10),规则 2 成立.规则 2 可以直接确定部分对象的区间的左端点,避免了搜索过程(算法 5 中的 Line 11~Line 12).规则 3 显然成立.更新区间左端点时,a 总是在 e 之前进行的,因此如果 $e\in DO(a).A$,则根据规则 3,$a.lower$ 可以用来缩小计算 e 的区间左端点时的搜索区间(算法 5 中的 Line 14~Line 16 所示).

引理 1. 对于 $P_{new}(e)\geqslant q$ 的对象 e,如果存在 $a\in C$,$a\prec e$,并且晚于 a 到达并支配 a 的对象都不属于 C,则 e 的区间左端点应该大于等于$\kappa(b_a)$.

证明:显然,如果 $a\in C$,那么 a 的后向关键祖先 b_a 肯定是存在的.因为 $a\in C$,所以 $P_{new}(a)<q$,并且可以推出$\kappa(e)>\kappa(b_a)$;不然,如果$\kappa(e)<\kappa(b_a)$(因为 e 不可能等于 b_a,所以$\kappa(e)\neq\kappa(b_a)$),则$\forall a'$,如果 $a'\prec a$,$\kappa(a')>\kappa(a)$(实际上,根据 b_a 的定义有$\kappa(a')\geqslant\kappa(b_a)$),那么 $a'\prec e$(因为 $a'\prec a$ 且 $a\prec e$)并且$\kappa(a')\geqslant\kappa(b_a)>\kappa(e)$.因此必然有 $P_{new}(e)\leqslant P_{new}(a)<q$,与条件矛盾.令 $P_{new}(a)=\prod_{i=1}^{m}(1-P(c_i))$,其中,$c_i$ 是 $S_{N,q}$ 中晚于 a 到达并且支配 a 的对象,且有$=c_1b_a,c_m=e_{new}$(e_{new} 是刚到达的对象).显然,所有$(1-P(c_i))(1\leqslant i\leqslant m)$都是计算 $P_{sky}(e)$的因子,并且 $c_i(1\leqslant i\leqslant m)$都属于 $S_{N,q}$,则如果在$[\kappa(b_a),\kappa(e_{new})]$范围内计算,必有 $P_{sky}(e)\leqslant\prod_{i=1}^{m}(1-P(c_i))<q$,如此,根据对象的区间左端点的定义(定义 10),可知 e 相应的区间

© 中国科学院软件研究所 http://www.jos.org.cn

左端点应该大于等于$\kappa(b_a)$. □

定理 5. 对于 $P_{new}(e)\geqslant q$ 的对象 e,如果存在 $a\in C, a\prec e$,则 e 的区间左端点应该大于等于$\kappa(b_a)$.

证明:如果 a 满足引理 1 的条件,则得证;如果 a 不满足引理 1 的条件,即存在晚于 a 到达并支配 a 的对象属于 C,则挑选这些对象中时间标签最大的,记为 a',则 a'满足引理 1 的条件.由 $a'\prec a$ 得$\kappa(b_{a'})>\kappa(b_a)$,则由引理 1 可知定理 5 成立. □

规则 4. 令 $k_e=\max\{\kappa(b_a)|a\prec e\wedge a\in C\}$,则根据定理 5 中的条件,$k_e$ 可以用来缩小确定对象 e 区间左端点时搜索的时间标签范围.

算法 5 的 Line 17 中,*find_dominator_in_C*()的作用就是确定这样的 k_e.算法 5 在应用了规则 4 之后,最后在限定的时间标签范围内($[\max\{l,k_e\},\kappa(a_e)]$)以深度优先模式在 $L(S_{N,q})$上搜索(算法 5 中的 Line 20)所有支配 e 的对象,然后再计算出 e 的区间左端点.

3 处理连续 *q*-skyline 查询的 PnNCont 算法

本文详细分析了新对象到达后 $SKY_{n,q}$ 变化,提出处理连续查询的 PnNCont 算法.令 e_{new} 为新到达的对象,M 为 e_{new} 到达前已经到达的对象个数.我们通过将 $S_{N,q}$ 中的对象分为下面 4 类分别讨论(区间变化包括从区间树中删除和更新左端点两种):

1) 属于 $SKY_{n,q}$ 且区间变化.其中,部分对象是因为 $P_{new}<q$ 而需要从 $SKY_{n,q}$ 删除;剩下的留在 $S_{N,q}$ 中对象的区间变化意味着区间左端点右移,此时需要判断$(M+1)-n+1$ 是否仍然包含在这些区间中,若不然则从 $SKY_{n,q}$ 删除;
2) 属于 $SKY_{n,q}$ 且区间没有变化.如果 $SKY_{n,q}$ 中包含了时间标签为 $M-n+1$ 的对象,那么将其删除;
3) 不属于 $SKY_{n,q}$ 且区间变化.区间左端点变化的幅度至少为 1,而当前刺入点$(M+1)-n+1$ 较 $M-n+1$ 只增加了 1,因此这些对象仍然不属于 $SKY_{n,q}$;
4) 不属于 $SKY_{n,q}$ 且区间无变化.令该类中对象 e 的区间为$(l,\kappa(e))$.因 $e\notin SKY_{n,q}$,则 $M-n+1\notin(l,\kappa(e))$.如果 $l=M-n+1$,那么刺入点变为$(M+1)-n+1$ 后,e 需要加入到 $SKY_{n,q}$.也就是说,所有以 $M-n+1$ 为区间左端点的对象都需要加入到 $SKY_{n,q}$.令 c 表示时间标签为 $M-n+1$ 的对象.如果 $P_{new}(c)<q$,则根据定理 5,$\kappa(c)$不可能成为某个对象的区间左端点,则必有 $P_{new}(c)\geqslant q$.进一步,如果 $P(c)P_{new}(c)\geqslant q$,则 e_{new} 到达之前 $c\in SKY_{n,q}$,否则仍有 $c\notin SKY_{n,q}$.但是为了方便处理,本文连续算法实现时总是将 c 加入到 $SKY_{n,q}$ 中(输出结果时可以很方便地判断 c 是否应该输出,因此不影响 $SKY_{n,q}$ 的处理).因此,根据 $SKY_{n,q}$ 中对象的最小时间标签值是否为 $M-n+1$,可以判断需不需要处理本类别(以及第 2 类)的对象.

本文采用红黑树[17]保存 $SKY_{n,q}$,$e_{\min}$ 是其中时间标签最小的对象.算法 6 描述了具体的连续查询过程.当新对象到达时,首先考虑第 1 类对象(Line 3~Line 8),将其中 $P_{new}<q$ 的对象和区间不再包含$(M+1)-n+1$ 对象从 $SKY_{n,q}$ 中删除,然后判断 e_{new} 是否属于 $SKY_{n,q}$(Line 9),最后判断是否处理第 2 类和第 4 类中的对象(Line 10~Line 13).第 3 类对象无需做处理.算法 6 中 *Add*()和 *Remove*()操作分别表示将对象加入到 $SKY_{n,q}$ 或则将对象从 $SKY_{n,q}$ 中删除.

算法 6. PnNCont 算法:处理 *n*-of-*N* 模型上的连续 *q*-skyline 查询.

描述:

```
1.  while 最新对象 e_new 到达 do
2.    M←M+1;
3.    for e∈SKY_{n,q}∩(DO(e_new)∪NR(e_new)) do
4.      for e'∈SKY_{n,q}∩DOTree(e) do        /*DOTree(e)包括 e*/
5.        if e'的区间中不包含 M–n+1 then Remove(e',SKY_{n,q}); end if
6.      end for
7.    end for
```

© 中国科学院软件研究所 http://www.jos.org.cn

8. **for** $e \in SKY_{n,q} \cap C$ **do** *Remove*(e,$SKY_{n,q}$); **end for**
9. **if** e_{new} 的区间包含 $M-n+1$ **then** *Add*(e_{new},$SKY_{n,q}$); **end if**
10. **while** $\kappa(e_{\min}) < M-n+1$ **do**
11. *Remove*($e_{\min}$,$SKY_{n,q}$);
12. **for** e:e 的区间左端点为$\kappa(e_{\min})$ **do** *Add*(e,$SKY_{n,q}$); **end for**
13. **end while**
14. **end while**

4 实验结果和分析

本节通过实验评估了算法性能.据本文作者所知,迄今没有发现针对概率流 n-of-N 模型上 q-skyline 查询而设计的算法,但理论上有一种简单方法可以直接应用于该问题,即对于每个 n 值以 DS_n 作为整个滑动窗口,用在概率流计算上计算整个滑动窗口上的 q-skyline 的算法(如 Zhang 等人[12]的方法)计算 DS_n 上的 q-skyline.此简单方法可以与本文方法对比,因此本文评估的算法有:

- 简单方法:上面提到的简单方法;
- 即时查询算法:即时查询算法就是区间树上的刺入查询算法;
- PnNM 维护算法:即维护候选集合 $S_{N,q}$ 以及区间树的算法 1;
- PnNCont 连续查询算法:即连续的计算 $SKY_{n,q}$ 查询结果的算法 6.

本文所有算法用 C 语言实现,用 GNU GCC 编译.运行实验的 PC 机内存 1G,CPU 是 AMD Atholon^{TM} 64 X2 Dual Core 3800+(2GHz),L2 Cache 为 512k.操作系统是 Ubuntu Linux 8.10.

本文的数据集为使用文献[1]中方法生成的数据集,这是在 skyline 研究中广泛使用的标准测试数据集.本文生成的测试数据集包含 200 万个数据对象(作为数据流中的对象),维度从 2~5,数据分布分别为独立(independent)、相关(correlated)和反相关(anti-correlated)的 3 类数据集.每个数据对象的出现概率为随机生成,服从(0,1)之间的均匀分布(uniform)或正态分布(normal).实验中滑动窗口大小 N 缺省为 10^6(1M),概率阈值(q)缺省为 0.3,维度(d)缺省为 4 维;出现概率缺省时服从(0,1)之间的均匀分布;当出现概率服从正态分布时,其均值为 0.5,标准差为 0.3;当出现概率服从均匀分布时,对于同一均值采用使方差最大的分布(此时分布区间最长,概率值可以在一个尽可能大的范围内出现).实验中,缺省的数据分布为反相关分布(因为下面的实验表明,反相关数据分布下处理开销是最大的).在测试算法时间开销时,因为一次查询的时间难以准确记录,因此实验中记录的是 1 000 次查询的平均处理时间.

我们首先阐述以上简单方法的低效率.Zhang 等人[12]的方法是计算固定长度概率流滑动窗口上 q-skyline 的一种高效方法,因此本文选取 Zhang 的方法作为简单方法中的计算方法.对于简单方法而言,如果同时提交多个不同 n 值的 q-skyline 查询,那么对每个 n 都要重新执行一次算法,也就是说,对每个 n 都要重新在 DS_n 上建立支持 q-skyline 查询的数据结构.对 Zhang 的方法而言,数据结构建立之后即可得到查询结果,所以重建数据结构的时间也就是查询时间.本文基于 1M 大小的滑动窗口、选取中等大小的 n 值(n=100k)比较本文的(即时)查询算法(ad hoc query)和简单方法(Naïve method)的效率,结果如图 4 所示(请注意,其中两幅子图的纵轴单位不同).

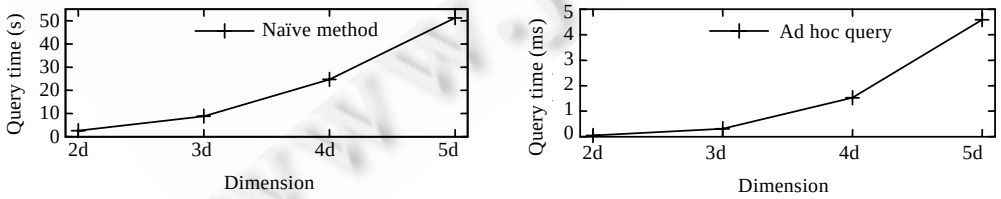


Fig.4 Query time of ad hoc query algorithm and the Naïve method

图 4 即时查询算法和简单方法查询时间对比

© 中国科学院软件研究所 http://www.jos.org.cn

从图 4 中可以看出,简单方法的查询时间至少几秒(当 d=2 时 2.62 秒),多则几十秒,如此长的查询时间根本不适合数据流环境.因此,本文下面不再与简单方法进行对比,而只评估本文算法的空间开销和时间开销.

4.1 即时查询算法的时间开销

本文的即时查询算法即是区间树上的刺入查询算法,其开销是 $O(d\log\log N+(\log n)^{d-1})$.因为反相关数据集是 3 种数据分布中处理开销最大的,为节省篇幅,本节及下面小节中只给出反相关数据分布下的实验结果.

图 5 中图 5(a)~图 5(d)这 4 个子图分别表明即时查询时间开销随维度增大而增加、随出现概率均值的增大而先增加后减少、随概率阈值的增大而减小以及随 n 值的增大而增加.

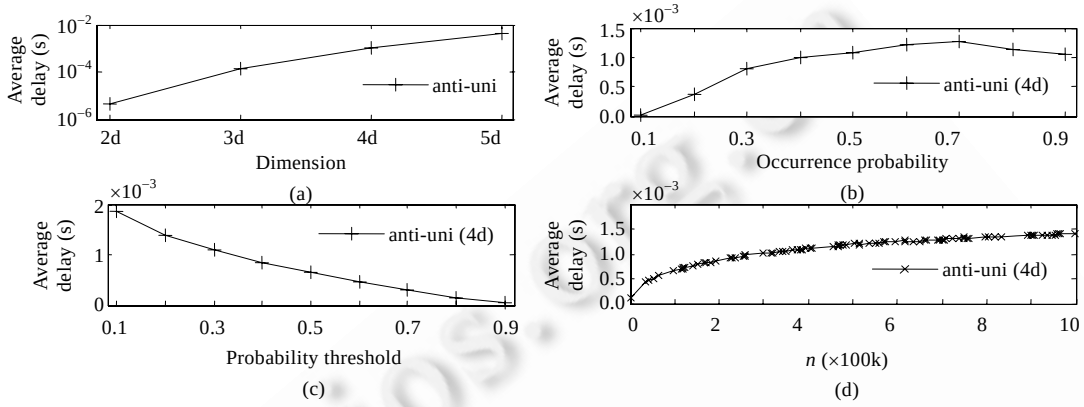


Fig.5 Average delays of the ad hoc query algorithm in various situations

图 5 各种情况下即时查询算法的时间开销

4.2 PnNM维护算法的时间开销

本节测试了 PnNM 维护算法的时间开销.从图 6 中可以看出,PnNM 维护算法的时间开销随着维度的增大而增加(图 6(a))、随着出现概率的增大而减少(图 6(b))、随着概率阈值的增大而减少(图 6(c))、随着滑动窗口的增大而增加(图 6(d)).

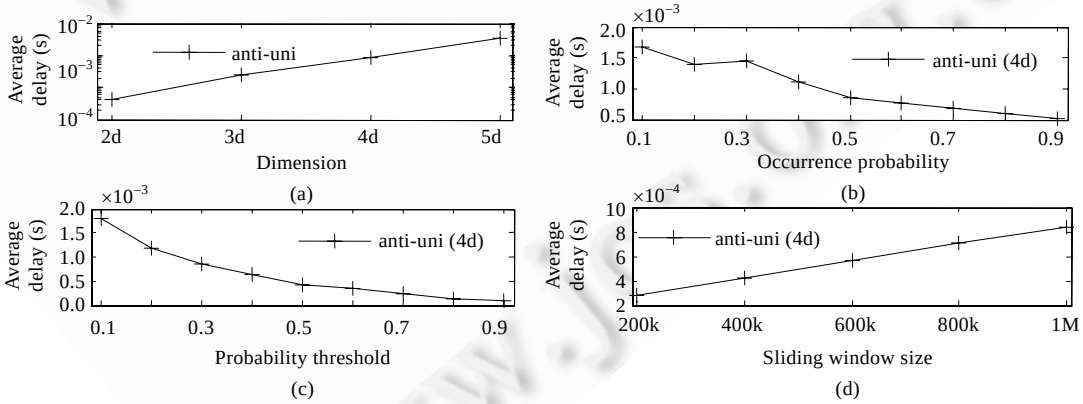


Fig.6 Average delays of the PnNM maintaining algorithm in various situations

图 6 各种情况下 PnNM 维护算法的时间开销

4.3 总体性能

本节评估概率数据流中对象不断到达时系统持续的处理性能.对象的处理时间是指连续两个对象之间的

© 中国科学院软件研究所 http://www.jos.org.cn

处理时间,包括前面对象到达后维护索引结构的时间和两个对象之间处理查询请求的时间.实验以生成的 2M 个数据对象作为数据流中的对象,窗口长度为 1M,窗口填满后开始滑动,同时开始每 1 000 个连续对象记录一次平均处理时间,结果如图 7 所示.

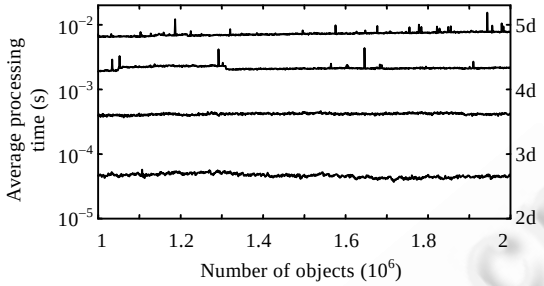


Fig.7 Overall performance

图 7 总体性能

图 7 表明,虽然在较高维(4d 和 5d)运行时偶尔出现时毛刺现象(因为 CPU 的 cache 较小),但系统的持续处理性能总体来说是很平稳的.数据为 2d 时可以处理高速数据流(超过每秒 10 000 个对象);最坏情况下,数据为 5d 时也可以处理中速数据流(超过每秒 100 个对象).

4.4 PnNCont连续查询算法的时间开销

本节评估 PnNCont 连续查询算法即算法 6 的时间开销,并比较算法 6 和使用每次运行即时(ad hoc)查询算法重新计算整个结果集的直接方法.因为连续查询算法中的部分维护工作已经集成到维护算法中(参见第 3 节),所以本节实验记录的时间是 1 000 次查询的平均处理时间.实验中滑动窗口长度为 1M,给出用 5 个 n 值(分别为窗口长度的 1/5,2/5,3/5,4/5 和全部)进行查询的结果,实验结果如图 8 所示.

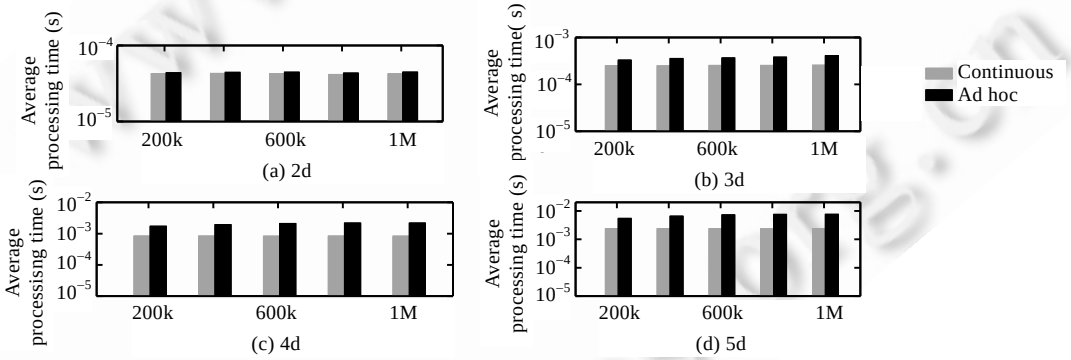


Fig.8 Continuous query algorithm vs. ad-hoc query algorithm under various n

图 8 不同 n 值下连续查询算法与即时查询算法的时间开销比较

图 8 展示了不同维度下连续查询算法和即时查询算法在处理连续查询时的时间开销对比.在较低维(2d 和 3d)时,性能提高的幅度比较高维(4d 和 5d)提高的幅度小,这是因为在较低维时 $SKY_{n,q}$ 较小,因此执行查询的时间占总的处理时间的比重很小,所以查询上提高的幅度就不明显.较高维时提高的幅度就很明显:处理 4d 数据时,数据流的速度可以从约每秒 490 个对象提高到约每秒 1 190 个对象;处理 5d 数据时,数据流速度可以从约每秒 140 个对象提高到约每秒 420 个对象.总而言之,图 8 表明,PnNCont 连续查询算法相对于简单地每次重新运行即时查询算法的方法,能够极大提高处理性能.

© 中国科学院软件研究所 http://www.jos.org.cn

5 结论和将来工作

本文研究了概率数据流上 *q*-skyline 查询问题,与已有的只支持滑动窗口模型的方法[12]相比,本文方法能够支持更为通用的 *n*-of-*N* 模型,并采用将 *q*-skyline 查询转换为区间树上刺入查询的方法支持 *n*-of-*N* 模型.提出了 PnNM 算法维护支持 *n*-of-*N* 模型所需的数据结构.为保证 PnNM 维护算法的高效率,还提出了基于 R 树的双层索引结构 RDO 树存储候选数据对象,在确定候选集合中被最新到达对象支配的对象时减小了搜索空间;然后在候选数据对象的时间标签上建立基于扩展红黑树的索引以高效更新区间;此外还利用概率数据流的特点提出一些启发式规则优化了区间更新过程.为保证 PnNM 算法的正确性,本文分析并保证了一些关键概念应用于不确定数据上的正确性,也分析了数据对象的不确定性后对算法正确性和复杂性的影响.在 PnNM 维护算法的基础上,本文使用区间树上的刺入查询算法作为即时查询算法.对于连续查询,本文并不每次重新计算一次结果,而是将连续查询结果集的计算看作一个连续更新的过程,分析了连续更新的特点并提出高效的 PnNCont 连续查询算法.最后,大量的实验结果表明,本文算法的是高效的,可以胜任快速概率数据流上 *q*-skyline 查询的处理.进一步的工作包括将给定概率阈值的连续查询扩展到同时支持多个概率阈值的连续查询.

References:

[1] Borzsonyi S, Kossmann D, Stocker K. The skyline operator. In: Ehrgott M, Greco S, Figueira J, eds. Proc. of the 17th Int'l Conf. on Data Engineering. Heidelberg: IEEE Computer Society, 2001. 421−430. [doi: 10.1109/ICDE.2001.914855]

[2] Chomicki J, Godfrey P, Gryz J, Liang D. Skyline with presorting. In: Dayal U, Ramamritham K, Vijayaraman TM, eds. Proc. of the 19th Int'l Conf. on Data Engineering. Bangalore: IEEE Computer Society, 2003. 717−719. [doi: 10.1109/ICDE.2003.1260846]

[3] Tan KL, Eng PK, Ooi BC. Efficient progressive skyline computation. In: Apers PMG, Atzeni P, Ceri S, Paraboschi S, Ramamohanarao K, Snodgrass RT, eds. Proc. of the 27th Int'l Conf. on Very Large Data Bases (VLDB 2001). Rome: Morgan Kaufmann Publishers, 2001. 301−310.

[4] Kossmann D, Ramsak F, Rost S. Shooting stars in the sky: an online algorithm for skyline queries. In: Bernstein PA, Loannidis YE, Ramakrishnan R, eds. Proc. of the 28th Int'l Conf. on Very Large Data Bases (VLDB 2002). Hong Kong: Morgan Kaufmann Publishers, 2002. 275−286.

[5] Chan CY, Eng PK, Tan KL. Stratified computation of skylines with partially-ordered domains. In: Ozcan F, ed. Proc. of the 2005 ACM SIGMOD Int'l Conference on Management of Data (SIGMOD 2005). Maryland: ACM Press, 2005. 203−214. [doi: 10.1145/1066157.1066181]

[6] Balke WT, Güntzer U, Zheng JX. Efficient distributed skylining for Web information systems. In: Bertino E, Christodoulakis S, Plexousakis D, Christophides V, Koubarakis M, Bohm K, Ferrari E, eds. Advances in Database Technology, Proc. of the 9th Int'l Conf. on Extending Database Technology (EDBT 2004). Heraklion: Springer-Verlag, 2004. 256−273. [doi: 10.1007/978-3-540-24741-8_16]

[7] Huang ZY, Jensen CS, Lu H, Ooi BC. Skyline queries against mobile lightweight devices in MANETs. In: Liu L, Reuter A, Whang K, Zhang J, eds. Proc. of the 22nd Int'l Conf. on Data Engineering (ICDE 2006). Atlanta: IEEE Computer Society, 2006. 66. [doi: 10.1109/ICDE.2006.142]

[8] Tao YF, Papadias D. Maintaining sliding window skylines on data streams. IEEE Trans. on Knowledge and Data Engineering, 2006, 18(3):377−391. [doi: 10.1109/TKDE.2006.48]

[9] Pei J, Jiang B, Lin XM, Yuan YD. Probabilistic skylines on uncertain data. In: Koch C, Gehrke J, Garofalakis MN, Srivastava D, Aberer K, Deshpande A, Florescu D, Chan CY, Ganti V, Kanne C, Klas W, Neuhold EJ, eds. Proc. of the 33rd Int'l Conf. on Very Large Data Bases (VLDB 2007). Vienna: ACM Press, 2007. 15−26.

[10] Lin XM, Lu HJ, Xu J, Yu J X. Continuously maintaining quantile summaries of the most recent *n* elements over a data stream. In: Proc. of the 20th Int'l Conf. on Data Engineering (ICDE 2004). Boston: IEEE Computer Society, 2004. 362−374.

[11] Lin XM, Yuan YD, Wang W, Lu HJ. Stabbing the sky: Efficient skyline computation over sliding windows. In: Proc. of the 21st Int'l Conf. on Data Engineering (ICDE 2005). Tokyo: IEEE Computer Society, 2005. 502−513. [doi: 10.1109/ICDE.2005.137]

© 中国科学院软件研究所 http://www.jos.org.cn

[12] Zhang WJ, Lin XM, Zhang Y, Wang W, Yu JX. Probabilistic skyline operator over sliding windows. In: Proc. of the 25th Int'l Conf. on Data Engineering (ICDE 2009). Shanghai: IEEE Computer Society, 2009. 1060−1071. [doi: 10.1109/ICDE.2009.83]

[13] Sun SL, Li JJ, Zhu YY. Efficient processing of continuous skyline query over distributed data streams. Journal of Software, 2009, 20(7):1839−1853 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/3340.htm [doi: 10.3724/SP.J.1001.2009.03340]

[14] Tian L, Zou P, Li AP, Jia Y. Grid index based algorithm for continuous skyline computation. Chinese Journal of Computers, 2008, 31(6):998−1012 (in Chinese with English abstract).

[15] Sun SL, Dai DB, Huang ZH, Zhang QX, Zhou LX. Algorithm on computing skyline over probabilistic data stream. Acta Electronica Sinica, 2009,37(2):285−293 (in Chinese with English abstract).

[16] Beckmann N, Kriegel HP, Schneider R, Seeger B. The R*-tree: An efficient and robust access method for points and rectangles. In: Garcia-molina H, Jagadish HV, eds. Proc. of the '90 ACM SIGMOD Int'l Conf. on Management of Data. Atlantic City: ACM Press, 1990. 322−311. [doi: 10.1145/93597.98741]

[17] Cormen TH, Leiserson CE, Rivest RL, Stein C. Introduction to Algorithms. 2nd ed., Cambridge: The MIT Press, 2001. 273−301.

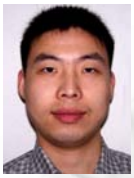
[18] de Berg M, Cheong O, vam Kreveld M, Overmars M. Computational Geometry: Algorithms and Applications. 3rd ed., Springer-Verlag, 2008. 220−225.

附中文参考文献:

[13] 孙圣力,李金玖,朱扬勇.高效处理分布式数据流上 skyline 持续查询算法.软件学报,2009,20(7):1839−1853. http://www.jos.org.cn/1000-9825/3340.htm [doi: 10.3724/SP.J.1001.2009.03340]

[14] 田李,邹鹏,李爱平,贾焰.基于网格索引的连续 Skyline 计算方法.计算机学报,2008,31(6):998−1012.

[15] 孙圣力,戴东波,黄震华,张齐勋,周力新.概率数据流上 Skyline 查询处理算法.电子学报,2009,37(2):285−293.



杨永滔(1981－),男,湖南湘潭人,博士生,CCF 学生会员,主要研究领域为不确定数据管理,数据流查询.

王意洁(1971－),女,博士,教授,博士生导师,CCF 高级会员,主要研究领域为网络计算,海量数据管理,虚拟化技术.

© 中国科学院软件研究所 http://www.jos.org.cn