概率数据流上 Skyline 查询处理算法

孙圣力[1],戴东波[2],黄震华[3],张齐勋[1],周立新[1]
(1. 北京大学软件与微电子学院,北京 102600;2. 复旦大学计算机科学技术学院,上海 200433;3. 同济大学电信学院,上海 200092)

摘 要: 概率数据流管理与分析逐步引起了研究者们的关注. Skyline 查询技术是近年来数据库领域的研究热点. 此前相关工作仅限于静态数据集或传统确定性数据流上的 Skyline 查询处理,尚无人考虑概率数据流上的 Skyline 计算问题,本文提出的 SOPDS 算法则较好地解决了该问题. 在采用适应性更强的网格索引的基础上,提出了概率定界、逐步求精、提前淘汰与选择补偿等启发式规则对算法从时间和空间两方面进行了系统地优化. 实验表明,算法在时间与空间上具有较高的整体性能.

关键词: 概率数据流; Skyline; 逐步求精; 提前淘汰
中图分类号: TP391.41 **文献标识码**: A **文章编号**: 0372-2112 (2009) 02-0285-09

Algorithm on Computing Skyline over Probabilistic Data Stream

SUN Sheng-li[1], DAI Dong-bo[2], HUANG Zhen-hua[3], ZHANG Qi-xun[1], ZHOU Li-xin[1]
(1. *School of Software and Microelectronics, Peking University, Beijing* 102600, *China*;
2. *School of Computer Science and Technology, Fudan University, Shanghai* 200433, *China*;
3. *School of Electronics and Information, Tongji University, Shanghai* 200092, *China*)

Abstract: Management and analysis of uncertain, probabilistic data stream has attracted considerable attention within database community. Skyline query processing is an open question recently. Although previous work has addressed skyline computations over static data or traditional data stream, skyline computation over probabilistic data stream is still at large. We propose an efficient algorithm SOPDS to handle this issue. Based on more adaptable grid index, a set of heuristic rules like probability bounding, progressive refinement, pre-elimination and selective compensation are developed to improve the comprehensive performance of SOPDS from point of reducing both CPU overhead and memory consumption. Massive experiments demonstrate that SOPDS is of high overall performance.

Key words: probabilistic data stream; skyline; progressive refinement; pre-elimination strategy

1 引言

多维空间上的 Skyline 查询[1]处理技术是近年来数据库领域的研究热点. Skyline 在偏好查询、多标准决策支持以及数据挖掘与可视化等方面应用广泛. 此前大量的工作都专注于在静态数据集上计算 Skyline[1,4,5],近年来也出现了一些在滑动窗口中计算 Skyline 的研究成果[2,3]. 最近以来,一种被称为概率数据流[15]的数据形态逐步引起了人们的关注. 以下给出一个概率数据流上 Skyline 查询的例子.

例 1. 存在某在线的电影评价数据库,把其中的每条评价记录看作一个对象,每个对象具有电影 ID、题材和表达手法等属性,后两者为评价指标分别记做 X 和 Y. 另给每个评价关联一个概率值 e,代表该评价的置信度. 评价以数据流的形式到达. 在下表 1 中,对象 u_i 代表第 i 个到达系统的评价记录. 现提出这样的查询:在最近到达的 10 个记录中找出至少以 0.3 的置信度保证不被其它电影支配的电影.

表 1 示例数据集

Object	u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8	u_9	u_{10}	u_{11}
X	0.40	0.95	0.48	0.22	0.65	0.43	0.58	0.64	0.62	0.16	0.37
Y	0.25	0.35	0.38	0.46	0.42	0.44	0.54	0.70	0.40	0.64	0.80
e	0.20	0.60	0.40	0.30	0.30	0.50	0.80	0.60	0.50	0.70	0.60

该查询的结果由图 1 所示. 在图 1(a) 中, u_6 属于 Skyline 的概率(简称 Skyline 概率)等于这样一个事件的概率: u_6 存在并且不存在支配 u_6 的对象,这一事件发生的概率为 $u_6.e * (1-u_1.e) = 0.4$. u_2, u_3, u_4 的 Skyline 概率分别为 0.48, 0.32 和 0.3. 而 u_1, u_5, u_7 和 u_8 的

收稿日期:2008-01-27;修回日期:2008-07-20
基金项目:国家 973 重点基础研究发展规划(No. 2005CB321905)

Skyline 概率均低于阈值 0.3. 故 u_8 到达后,窗口上的 Skyline 由 u_2, u_3, u_4 和 u_6 组成(用实心点表示).

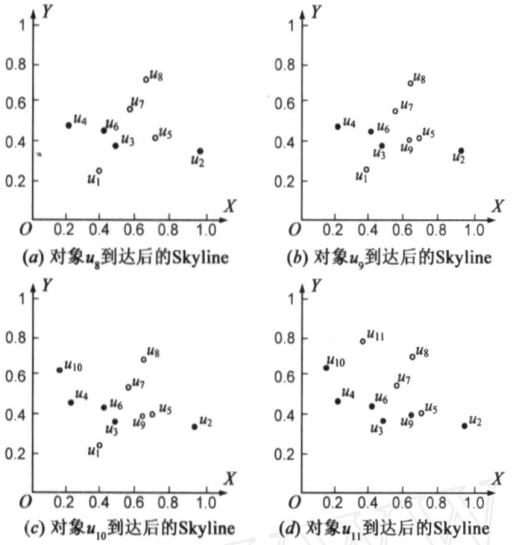


图1 概率数据流上的Skyline演化

其他情况不再冗叙. 由本例可见,与传统确定性数据流上的 Skyline 计算[2]不同,概率数据流上的计算难点是:(1)尽可能早地淘汰那些不再有机会加入 Skyline 的对象;(2)采用合适的策略高效地确定新到达对象的身份. 本文的贡献是:提出了一个高效地计算概率数据流上 Skyline 的算法 SOPDS,并开发了多个启发式方法显著地提高了算法的时间与空间性能. 大量实验表明本文提出的算法相当高效.

2 背景知识

2.1 相关工作

数据流管理与分析[7,17]是近年来数据库领域的研究热点. 最近两年以来,数据流的研究重心逐步转移到不确定的概率数据流之上[8,9,14,15]. 数据的不确定性分为两种:元组不确定性和属性不确定性. 元组不确定性是指关系数据库中的一个元组关联着一个概率值表明其存在的可能性[9~11]. 本文的数据模型即是基于元组不确定性的. 属性不确定性是指元组中的每个属性值是不精确的,精确程度以一定概率(或概率密度函数)表示[12,13]. 文献[9~13]主要研究概率数据在静态的不确定数据库中如何保持模型语义的完整性以及查询的高效性;文献[8,14,15]则进一步研究在数据流的环境下如何经一遍扫描有效地计算概率数据流上的聚集结果,如:F_0、F_1、F_2、平均值(mean)、中位点(median)、最小值和最大值等.

Skyline 的定义最早由 Borzsonyi 等人在文[1]中提出. 此后涌现出大量适用于不同场合的更高效的算法,比较有代表性的是基于索引结构的 *BBS* 算法[4]和基于排序的 *SFS* 算法[5]. 与本文问题相近的工作有:确定性数据流上的 Skyline 查询处理[2,3]和静态不确定数据集上的 Skyline 计算[6]. 文献[6]考虑的数据模型为数据集由若干确定的对象组成,而每个对象具有数目不等的若干实例. 以各对象的实例的 Skyline 概率为基础提出了概率 Skyline 的概念. 文献[6]考虑的数据模型实质上是属性不确定型的离散化版本,而本文考虑的则是元组不确定性的情形. 另一方面,文献[6]考虑的是静态数据集而本文考虑的是数据流. 文献[2,3]考虑如何高效地维护确定性数据流上的 Skyline. 在第 5 节的实验中,将文献[2]中的算法直接扩展到概率数据流环境中,将其作为基准算法与 *SOPDS* 算法进行性能对比. 对比实验表明,无论是时间还是空间性能 *SOPDS* 算法都具有压倒性优势.

2.2 术语与问题定义

令 $\mathrm{A}=\{A_1, A_2, \cdots, A_D\}$ 是一组有界并有序的域,$\mathrm{O}=A_1\times A_2\times\cdots\times A_D$ 是一个 D 维空间. 称 $A_1, \cdots, A_D$ 为空间O 的属性或维. 不失一般性,本文假设 $\forall A_i\in\mathrm{A}$ 的定义域均为实数区间(0,1]. 考虑一个数据集 U,任何对象 $u\in U$ 均来自空间O ,且对象间相关独立. 将 u 在属性 A_i 上的取值记为 $u.val_i$.

定义 1 (支配). 给定任意两个对象 $u, v\in U$. 如果对于 $\forall A_i\in\mathrm{A}$,有 $u.val_i\le v.val_i$,且 $\exists A_j\in\mathrm{A}$,有 $u.val_j<v.val_j$,则说 u 支配 v,记为 $u\prec v$. 若 u 不支配 v,则记为 $u\not\prec v$.

定义 2 (Skyline). 集合 U 中所有不被其他对象所支配的对象组成的集合,称作 U 上的 Skyline,记为 $SKY(U)$. 形式化地,$SKY(U)=\{u\in U\mid \forall v\in U: v\not\prec u\}$.

与此前概率数据流上的相关工作[8]相一致,本文定义概率数据流为不确定元组的序列:$\{<u_1, p_1>, \cdots, <u_k, p_k>\cdots\}$,其中:$<u_i, p_i>$ 是一个不确定元组;u_i 是一个来自空间O 的对象,对象间相互独立;p_i 是对象存在的概率$(0<p_i\le 1)$;$i\,(i\ge 0)$ 代表元组(或对象)在数据流中的序列号. 本文考虑基于计数的滑动窗口模型[7],不妨假设滑动窗口的宽度为 W,即每个对象的生命周期跨度为 W 个序列号. 也就是说 u_i 在序列号区间 $[i, i+W-1]$上是有效的(或称为活动的),也称该区间为 u_i 的生命周期,当第 $i+W$ 个对象到达系统后 u_i 过期. 从另一个角度来说,还意味着只对数据流上最近的 W 个元组感兴趣. 设最近到达的对象为 u_N,则我们只考虑元组集合$\{<u_{N-W+1}, p_{N-W+1}>, \cdots, <u_{N,p_N}>\}$. 将该集合记为(,将其中的序列号集合$\{N-W+1, \cdots, N\}$记为 S. 则任意对象 u_i 的 Skyline 概率在数值上等于这样一个事件的概率:u_i 存在并且在 $u_i.ADR$(反支配域)

内不存在任何活动对象. 假设 $i \leq N \leq i + W - 1$,先定义一个序列号子集 $S' = \{j \mid < u_j, p_j > \in S \wedge u_j \prec u_i \wedge N - W + 1 \leq j \leq N\}$,显然 $S' \in S$. 这样 u_i 的 Skyline 概率由式(1)表示.

$$P_{SKY}(u_i) = p_i \cdot \prod_{j \in S'} (1 - p_j) \quad (1)$$

在滑动窗口模型下,Skyline 往往会随着不确定元组的到达与过期而演化. 本文要解决的问题描述为:给定一个概率阈值 $p(0 < p \leq 1)$,随着概率数据流中不确定元组的不断到达与过期,动态地维护(上的 $SKY_p(S)$. 对于基于时间戳的数据流模型,本文提出的算法只要稍加修改就能直接适用(具体见 4.3 节所述). 为了描述的简便,以下假设元组的存在概率为 $0 < p_i < 1$,对于 $p_i = 1$ 的情形算法也只要稍加修改就能直接适用(修改见 4.3 节所述). 表 2 给出本文常用的符号及其意义.

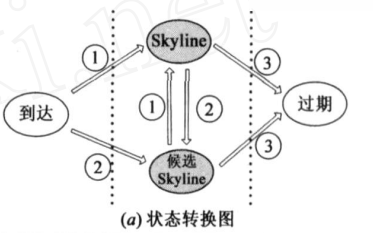
表 2　本文所用的符号说明

符号	描述
W	滑动窗口的宽度
D	对象的维度
S	基于概率数据流的滑动窗口
$SKY(U)$	确定数据集 U 上的 Skyline
$P_{SKY}(u)$	对象 u 的 Skyline 概率
$P^+_{SKY}(u)$	u 的 Skyline 概率的上界
$P^-_{SKY}(u)$	u 的 Skyline 概率的下界
$u.pbf$	u 不被它之前到达的活动对象支配的概率
$u.paf$	u 不被它之后到达的活动对象支配的概率
$SKY_p(S)$	滑动窗口上阈值为 p 的 Skyline

3　SOPDS 算法结构

本节先分析概率数据流环境下对象的状态,再阐述算法中所用的数据结构及算法的体系结构. 在设计阶段将概率数据流中的每个元组看作一个对象,记为:$u(u.id, u.e, u.val_1, \cdots, u.val_D)$,其中 $u.id$ 表示其在数据流中的序列号,$u.e$ 表示其存在的概率,$u.val_i$ 表示在属性 A_i 上的取值. 在概率数据流环境下对象呈现出 4 种不同的状态,即:到达、候选 Skyline、Skyline 和过期状态. 数据流中观察到的对象在进入系统之先保存在缓冲区中,对象位于缓冲区中,称之处于到达态;进入系统的对象,当前若不是 Skyline 对象但在将来有可能成加入 Skyline,称之处于候选 Skyline 态;当前是 Skyline 中对象的称之处于 Skyline 态;对象过期后称之处于过期态. 对象在其整个生命周期内呈现不同的状态,但在一个具体的时刻它只能处于一个确定的状态. 状态转换关系由图 2(a)所示,转换由一特定条件触发,触发条件由图 2(b)所示. 对象一旦过期立即从内存中淘汰.

与文献[16]类似,本文提出的 SOPDS 算法采用网格作为索引结构. 不失一般性,图 3 给出空间维度为 2 时算法的数据结构,此后的阐述将专注于空间维度为 2 时的情形. 采用队列来保存活动对象,将新到达的对象插入到队列尾端,而过期的数据对象从队列头端直接删除. 称该队列为活动对象列表,简记为 AOL. AOL 中对象的序列号是由头向尾递增的. 换一种说法,称较早到达系统的对象较"老",较迟到达的对象较"年轻",则 AOL 头端对象较"老",尾端对象较"年轻". 采用网格作为活动对象的索引,每个格关联 2 个指针列表,分别用来保存指向该格中的 Skyline 和候选 Skyline 对象的指针. 格中的每一个指针列表也是一个队列,指针列表随着数据对象的更新而动态维护,确保该队列所关联的对象的序列号也由头部向尾递增. SOPDS 算法的输出则由 Skyline 对象指针列表表现出来.



(a) 状态转换图

条件	描述
1	该对象的Skyline概率大于等于阈值p
2	该对象的Skyline概率小阈值p
3	该对象过期

(b) 触发条件

图2　状态转换图及触发条件

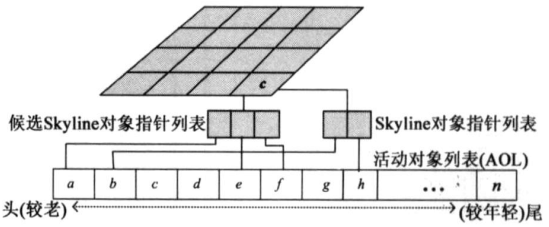


图3　SOPDS算法中的数据结构

SOPDS 主要由三个模块组成,算法各模块的合作情况简述如下:新到达的对象(设为 u)缓存在缓冲区 BF 中. 新对象到达后立即调用模块 $procExpObj$ 处理过期的对象(设为 v),包括从系统中淘汰过期的 v 并增大被 v 支配的对象的 Skyline 概率. 接着调用模块 $insertIncomObj$ 计算 u 的 Skyline 概率并将 u 插入到其所属格中相应的队列中. 最后调用模块 $procDomObj$ 处理所有被 u 支配的对象,即降低被 u 支配的对象的 Skyline 概率. 第 4 节将详细地阐述如何对 SOPDS 算法从时间与空间上进行优化.

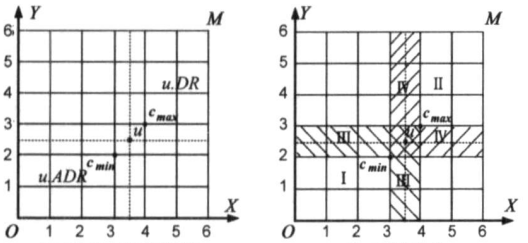
4　算法优化与实现

4.1　计算优化策略

减少计算复杂度的关键在于减少计算过程中支配测试的次数. 设新到达的对象为 u,将 $u.DR$(支配域)与

u.ADR 覆盖的格划分为 Ⅰ、Ⅱ、Ⅲ和 Ⅳ区,这些区域由图 4(b) 所示. 其中 Ⅰ、Ⅱ区分别由被 u.ADR 和 u.DR 完全覆盖的格组成,Ⅲ、Ⅳ区分别由被 u.ADR 和 u.DR 部分覆盖的格组成,u 本身所在的格既属于 Ⅲ区也属于 Ⅳ区. 这样,计算过程就只需在此四个区域中进行:计算新到达对象的 Skyline 概率只需考虑 Ⅰ、Ⅲ区中对象;处理被新来对象所支配的对象以及过期处理只需考虑 Ⅱ、Ⅳ区. 网格索引为优化措施的进一步开发提供了基础. 根据网格的特点,本文提出了概率定界、逐步求精等启发式规则来提高确定对象身份的效率.

启发式规则 1. 概率定界 考察新到达的对象 u,记格 c(对象 u 所处的格) 的右上角和左下角分别为 c_{max}和 c_{min},它们和 u 之间的关系由图 4(a) 所示. 显然,任何支配 c_{min}的对象必支配 u,而任何支配 u 的对象也必支配 c_{max}. 我们将格也看作一个对象,简称为格对象,除坐标属性外为每个格对象 c 新增加两个属性,$c.plb$ 和 $c.prt$. 其中 $c.plb$ 代表 c_{min}不被任何活动对象支配的概率,$c.prt$ 代表 c_{max}不被任何活动对象支配的概率. 以下给出 $c.plb$ 和 $c.prt$ 的形式化定义. 假设 $u.id \le N \le u.id + W - 1$,定义两集合 $V = \{v \mid N-W+1 \le v.id \le N \wedge v \prec c_{min}\}$, $W = \{w \mid N - W + 1 \le w.id \le N \wedge w \prec c_{max}\}$,则 $c.plb = \prod_{v \in V}(1 - v.e)$, $c.prt = \prod_{w \in W}(1 - w.e)$. 根据以上分析,得到如下定理.



(a) 对象与其所在格 (b) 区域划分

图4 计算优化策略示意图

定理 1 对象 u 在其生命周期内,$c.prt * u.e \le P_{SKY}(u) \le c.plb * u.e$ 总成立.

这样在计算 $P_{SKY}(u)$ 之时,可以令其下界 $P^{-}_{SKY}(u)$ 和上界 $P^{+}_{SKY}(u)$ 分别等于 $c.prt * u.e$ 和 $c.plb * u.e$. 算法此时仍然只得到一个 $P_{SKY}(u)$ 的估计值. 基于式(2)、(3) 和(4),$P_{SKY}(u)$ 与阈值 p 存在着三种可能的关系. 如果式(2) 成立,则必有 $P_{SKY}(u) \ge p$,此时 u 为 Skyline 对象;如果式(3) 成立,则必有 $P_{SKY}(u) < p$,u 为候选 Skyline 对象;如果式(4) 成立,则 u 身份未明需要进一步对概率求精. 在此引入符号 $P^{*}_{SKY}(u)$,令 $P^{*}_{SKY}(u) = P^{+}_{SKY}(u)$,故有 $P^{-}_{SKY}(u) \le P_{SKY}(u) \le P^{*}_{SKY}(u)$.

$$\begin{cases} P^{-}_{SKY} \ge p & (2) \\ P^{+}_{SKY} < p & (3) \\ P^{-}_{SKY} < p \le P^{+}_{SKY} & (4) \end{cases}$$

启发式规则 2. 逐步求精 如果式(4) 成立,则需要遍历 Ⅲ区对 $P_{SKY}(u)$ 求精以进一步确认 u 的身份. Ⅲ区中的对象与 u 之间的关系不能直接判别,对 Ⅲ区对象的访问意味着与 u 的一次支配测试. 逐步求精的思想即是在达到目的的前提下尽可能地减少访问的对象数,以减少支配测试的次数. 在实施过程中,对 Ⅲ区的访问采取逐格逐对象推进的方式进行,思想源于如下观察:若 $P_{SKY}(u) \ge p$,对 Ⅲ区的完全遍历不可避免;若 $P_{SKY}(u) < p$,则对 Ⅲ区中对象的访问存在剪枝的余地. 因为随着遍历的推进求得 $P^{*}_{SKY}(u)$ 值只会越来越小,如果算法已得到 $P^{*}_{SKY}(u) < p$,则没有继续向下推进的必要. 因为 $P_{SKY}(u) \le P^{*}_{SKY}(u)$ (继续遍历仍然可能找到支配 u 的对象,故成立),可以推断 $P_{SKY}(u) < p$,u 的身份已经明确,故求精可以就此中止.

为便于在算法实现中应用该规则,为对象 u 增加一个属性(bp),该属性用来保存在求精的过程中遍历停止的位置(该位置由格坐标以及该格中已被访问的对象的最大序列号组成,该位置也被称为断点). 断点的用处是:在 *InsertIncomObj* 中应用逐步求精策略后,算法得到往往只是 $P_{SKY}(u)$ 的估计值. 当支配 u 的对象过期或者到来支配 u 的较年轻的对象时,可能又会导致 u 的身份不明,即出现 $P^{-}_{SKY}(u) < p \le P^{*}_{SKY}(u)$. 这样就需要进一步求精,而断点则为进一步求精保留了计算的起点.

4.2 空间优化策略

空间优化策略考虑尽可能早地从滑动窗口中淘汰那些不再有机会加入 Skyline 的对象. 在保证算法的正确性与完备性的前提下,究竟哪些活动对象可以提前从系统中淘汰?显然该对象(u) 要同时满足以下两个条件:(1). u 没有机会再加入 Skyline;(2). 将 u 提前淘汰不对 $u.DR$ 中对象的身份判定造成影响.

首先对条件(1) 进行分析. 由式(1) 可知,$P_{SKY}(u)$ 实际上为 u 本身存在的概率与 u 不被其它任何活动对象所支配的概率之积. 而后者可以进一步分解为其不被较老的活动对象支配的概率与不被较年轻的活动对象支配的概率两部分. 为 u 新增两个属性 pbf、paf,分别保存这两部分概率. 形式化地,u 的生命周期为 $[u.id, u.id + W - 1]$,设 $u.id \le N \le u.id + W - 1$,定义两个集合 $R = \{r \mid r \prec u \wedge N - W + 1 \le r.id < u.id\}$, $S = \{s \mid s \prec u \wedge u.id < s.id \le N\}$. 则 $u.pbf$ 与 $u.paf$ 表示为:

$$\begin{cases} u.pdf = \prod_{r \in R}(1 - r.e) & (5) \\ u.paf = \prod_{s \in S}(1 - s.e) & (6) \end{cases}$$

这样 $P_{SKY}(u)$ 还可以表示为 $P_{SKY}(u) = u.e * u.pbf * u.paf$. 假设 u 当前是候选 Skyline 对象,在滑动窗口

环境下,u 在过期之前是否还有机会加入 Skyline 实际上是由 $u.paf$ 决定,下面给出引理.

引理 1　活动对象 u 在其生命周期内不再有机会再加入 Skyline 的充分必要条件是 $u.paf * u.e < p$.

证明:对象 u 的生命周期为 $[u.id, u.id + W - 1]$. 假设某对象 v 到来,$v.id \in \{u.id + 1, \cdots, u.id + W - 1\}$ 且 $v \prec u$,使得 $u.paf * u.e < p$. 则 $P_{SKY}(u) = u.e * u.pbf * u.paf < p$,而 u 在其余下的生命周期 $[v.id, u.id + W - 1]$ 内,$u.pbf$ 最大为 1. 因此 u 在其余下的生命周期内 $P_{SKY}(u) < p$,故其不再有机会加入 Skyline,充分性得证. 以下再证必要性. 假设 u 在其生命周期内 $u.paf * u.e \geq p$ 总是成立. 因为支配 u 的比 u 老的对象最迟会在第 $u.id + W - 1$ 个对象到达时过期,则至少在第 $u.id + W - 1$ 个对象到达而第 $u.id + W$ 个对象尚未到达的这段时间内 $u.pbf = 1$,至少在这段时间内 $P_{SKY}(u) = u.e * u.pbf * u.paf \geq p$,故 u 在其生命周期内仍有机会加入 Skyline. 必要性得证,综上所述,引理成立.

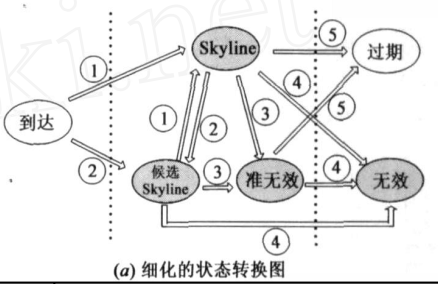
再分析条件(2),该条件实际上要求:对于 $u.DR$ 中的任意对象 v,若 $P_{SKY}(v) \geq p$,则算法要保证使得 $P^{*}_{SKY}(v) \geq p$;若 $P_{SKY}(v) < p$,则算法也要保证使得 $P^{*}_{SKY}(v) < p$. 以下给出定理.

定理 2　对于活动对象 u,若 $u.paf < p$ 成立,则可以安全地将其从系统中提前淘汰.

证明:对象 u 的生命周期为 $[u.id, u.id + W - 1]$. 假设因某对象 w 的到来,$w.id(\{u.id + 1, \cdots, u.id + W - 1\}$ 且 $w \prec u$,使得 $u.paf < p$ 成立. 显然此时 u 已经满足了引理 1 中的条件,以下主要证明其是如何满足条件(2)的. 此时(系统中最年轻对象为 w)定义集合 $V = \{v' \mid u.id < v'.id \leq w.id \wedge v' \prec u\}$,则 $u.paf = \prod_{v \in V}(1 - v', e)$,设 V 中最老的对象为 v. 能不能将 u 从系统中除去,关键看是否会对 $u.DR$ 中对象的身份确定造成影响. 将 u 提前淘汰只会影响区间 $[w.id - W + 2, u.id + W - 1]$ 上到达且被 u 支配的对象,将这些对象基于其序列号划分为三个集合:$R = \{r \mid w.id < r.id \leq u.id + W - 1 \wedge u \prec r\}$,$O = \{o \mid u.id < o.id < w.id \wedge u \prec o\}$,$Q = \{q \mid w.id - W + 2 \leq q.id < u.id \wedge u \prec q\}$. 先考察 R. 对于 $\forall r \in R$,v 过期之前 $P_{SKY}(r) < p$ 总成立(因为 r 被 V 中全部对象所支配,且 V 中全部对象此时都还是活动对象,故 $r.pbf \leq \prod_{v \in V}(1 - v', e) < p$.). 如果在 w 到来之后算法已经立即将 u 从系统中删除,则只要 v 仍然存在于系统中就一定也会使 $P^{*}_{SKY}(r) < p$(因为 $r.pbf < p$,故可保证). 而当 v 过期时,u 已经在此之前过期了. 这样 u 的提前淘汰不会影响 r 的身份确定. Q 与 O 中的对象也与此类似不再冗述. 故 $u.paf < p$ 成立时可以安全地将 u 从系统中除去,定理得证.

根据引理 1 与定理 2,可以将图 2(a)中的候选 Skyline 状态进一步细分. 若对象满足 7、8 或 9 式中的条件,则称其分别处于无效、准无效和候选 Skyline 状态. 位于这三个状态的对象分别称为无效、准无效和候选 Skyline 对象. 无效与准无效对象在不能再加入 Skyline,而前者可以安全地从系统中提前淘汰. 细化后的状态转换图及触发条件由图 5 给出. 总结以上分析,给出启发式规则 3.

$$\begin{cases} u.paf < p & (7) \\ u.paf \geq p \text{ 且 } u.paf * u.e < p & (8) \\ u.paf * u.e \geq p \text{ 且 } u.pdf * u.paf * u.e < p & (9) \end{cases}$$



(a) 细化的状态转换图

条件	描述
1	该对象的 Skyline 概率大于等于阈值 p
2	该对象满足(12)的条件
3	该对象满足(11)的条件
4	该对象满足(10)的条件
5	该对象过期

(b) 触发条件

图 5　细化的状态转换图及触发条件

启发式规则 3. 提前淘汰　对于活动对象 u,若条件 $u.paf < p$ 满足,则可以立即将其从系统中提前淘汰.

应用启发式规则 3 将对象 u 提前淘汰时,与正常的过期处理(由 *procExpObj* 实施)流程一样需要恢复环境,即:对于系统中 $\forall v \in \{v \mid u \prec v' \wedge v'.id < u.id\}$,$v.paf := v.paf / (1 - u.e)$;对于系统中 $\forall w(\{w \mid u \prec w' \wedge w'.id > u.id\}$,$w.pbf := w.pbf / (1 - u.e)$. 环境恢复优化方法由规则 4 给出.

启发式规则 4. 选择补偿　提前淘汰对象 u 时,只需对系统中的 $\forall w \in \{w \mid u \prec w' \wedge w'.id > u.id\}$ 进行概率补偿.

假设 u 被 r 提前淘汰,淘汰 u 时,$u.paf < p$. 对于 $(v(\{v \mid u \prec v' \wedge v'.id < u.id\}$,有 $v.paf \leq u.paf < p$. v 随后必被 r 提前淘汰,故 u 没有必要对这些对象进行概率补偿. 将这种策略称之为选择性补偿,简称为选择补偿.

4.3　算法实现与描述

SOPDS 算法主程序由图 6 给出,算法的主要模块

procExpObj、*insertIncomObj* 和 *procDomObj* 分别由图 7、8 和 9 描述. 另外子模块 *refineProb* 的功能是在 III区中对输入 u 的 Skyline 概率逐步求精,当 $u.pbf * u.paf * u.e < p$ 或 III区遍历完毕时返回;模块 *dropHopelessObj* 的则删除对象 u 并恢复环境. 此两子模块的详细描述分别在图 10、11 中给出.

```
Algorithm SOPDS//Skyline Over Probabilistic Data Stream
1  procExpObj;//处理过期的对象;
2  for BF 中的对象 u
3     iinsertIncomObj(u);//计算新来对象的 Skyline 概率
4     procDomObj(u);//处理被新来对象支配的对象
```

图 6 *SOPDS* 算法主程序

```
Procedure procExpObj
1  for AOL 中过期的对象 u
2  for u 的 II区与 IV区中每一格 c
3     e.prt := c.prt/(1 - u.e);
4     if c 位于 II区  e.plb := c.plb/(1 - u.e);
5     for c 中的每一对象 v
6        if c 位于 II区 or (c 位于 IV区 and u 位于断点 v.bp 之间 and
         u dominate v)
7           v.paf := v.pbf/(1 - u.e);
8           if v 来自候选 Skyline 列表
9              P+SKY(v) := c.plb * v.e; P-SKY(v) := c.prt * v.e;//以
                                                下 3 句体现规则 1
10             if P-SKY(v) ≥p
11                将 v 迁入到 e 中的 Skyline 列表;
12             else if P+SKY(u) ≥p;
13                refineProb(v);
14                if v.pbf * v.paf * v.e ≥p
15                   将 v 迁入到 c 中的 Skyline 列表;
```

图 7 处理过期对象过程

```
Procedure insertIncomObj
//输入为对象 u. c 为对象 u 所属的格.
1  u.pbf := c.plb;  //以下 6 句体现规则 1
2  P+SKY(u) := c.plb * u.e; P-SKY(u) := c.prt * u.e;
3  if P-SKY(u) ≥p
4     将 u 插入到 c 中的 Skyline 列表;
5  else if P+SKY(u) < p
6        将 u 插入到 c 中的候选 Skyline 列表;
7     esle
8        refineProb(u);
9        if u.pbf * u.paf * u.e ≥p
10          将 u 插入到 c 中的 Skyline 列表;
11          else
12             将 u 插入到 c 中的侯选 Skyline 列表;
```

图 8 确定新到对象身份的过程

本文此前假设对象存在概率位于区间 $(0,1)$. 对于概率为 1 的情形,算法只需在现有基础上略加修改就能适应. 为 u 再增加一个辅助属性 $u.temp$,其作用为:计算 $u.pbf$ 的过程时若遇到存在概率为 1 的对象 (v),则将此时的 $u.pbf$ 值保存到 $u.temp$ 中,待 v 过期后再将 $u.pbf$ 恢复为 $u.temp$. 对格对象 c 也作同样处理. 最后再对图 7 的第 3、4 和 7 句,图 10 的第 5 句作适当的修改就能适应概率为 1 的情形,具体不再冗述. 最后讨论 SOPDS 算法在基于时间戳的滑动窗口模型上的适应性问题. SOPDS 的核心是 4 个启发优化方法,算法的适应性问题取决于此优化方法的适应性问题. 只需将 $u.paf$ 的定义扩展为 u 不被同龄及更年轻的活动对象支配的概率,则所有启发优化方法均适用于基于时间戳的模型,故算法完全适用于基于时间戳的滑动窗口模型.

```
Procedure procDomObj
//输入为对象 u.
1  for u 的 II区与 IV区中每一格 c  //分层遍历
2  c.prt := c.prt * (1 - u.e);
3  if c 是 II区中的格
4     c.plb := c.plb * (1 - u.e);
5  for c 中的每一对象 v
6     if c 是 II区中的格 or (c 是 u 的 IV区中的格 and u dominate
      v)
7        v.paf := v.paf * (1 - u.e);
8        if v.paf < p  //以下 2 句体现规则 3
9           dropHopelessObj(v);
10       else
11          if v 来自 Skyline 列表
12             P+SKY(v) := c.plb * v.c; P-SKY(v) := c.prt * v.
                                          c;//以下 3 句体现规则 1
13          if P+SKY < p
14             将 v 迁入到 c 中候选 Skyline 列表
15          else if P-SKY(v) < p
16                refineProb(v);
17                if v.pbf * v.paf * v.e < p;
18                   将 v 迁入到 c 中候选 Skyline 列表;
```

图 9 处理被新到对象支配的对象过程

```
Procedure refineProb
//输入为对象 u
1  if u.pbf * u.paf * u.e < p return;
2  for u 的 III区中 u.bp 所在及其之后的每一格 c//以下 8 句体现
                                                规则 2
3  for c 中的每一对象 v//若 c 为 u 本身所在的格则从 u.bp 所指之
                                  后的对象开始计算
4     if c.id < u.id
5        if v dominate u
6           u.pbf := u.pbf * (1 - v.e);
7           if u.pbf * u.paf * u.e < p
8              将 v 的位置记入 u.bp;//保存断点
9              return;
```

图 10 概率求精过程

```
Procedure dropHopelessObj
//输入为对象 u
1  for u 的 II区与 IV区中每一格 c//分层遍历
2    for c 中的每一对象 v
3      if v.id > u.id  //以下 3 句体现规则 4
4        if c 是 II区中的格 or(c 是 IV区中的格 and u dominate v
           and u 位于断点 v.bp 之前)
5          v.pbf := v.pbf/(1 - u.e);
6  将 u 从系统中淘汰;
```

图 11　删除无效对象并恢复环境过程

4.4　算法复杂度分析

本节对 SOPDS 算法的时间复杂度进行简要地分析. 算法处理一个对象(假设为 u)的开销由 3 个模块 *procExpObj*、*insertIncomObj* 和 *procDomObj* 发生开销组成. 算法中的基本操作是支配测试,将支配测试的平均代价记为 C_{dom}. 设单个格中的平均对象数目为 α,而处理对象 u 时调用 *procDomObj*、*procExpObj* 引发其它对象身份变化的平均次数为 M. III、IV区包含的格的总数为 $K^D-(K-1)^D$. 确定 u 的身份产生的开销为 $O((K^D-(K-1)^D)\cdot\alpha\cdot C_{dom}+\alpha)$,前项为遍历 III区产生的开销,后项为将 u 插入到其所属格中合适的位置而产生的开销. 而调用模块 *procDomObj*、*procExpOb* 产生的开销的上界均为 $O((K^D-(K-1)^D)\cdot\alpha\cdot C_{dom}+M\cdot\alpha+(K-1)^D\cdot\alpha)$,第一项为遍历 IV区产生的开销,第二项是使其它对象身份发生变化而产生的开销,第三项是遍历 II区产生的开销. 综上所述,下面给出定理.

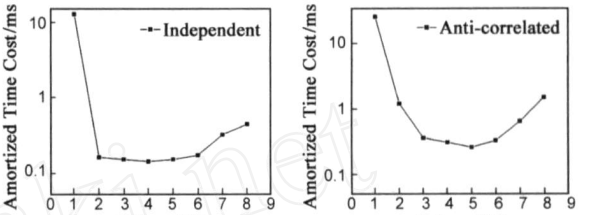
定理 3　SOPDS 算法处理一个对象平均分摊的时间开销为 $O((K^D-(K-1)^D)\cdot\alpha\cdot C_{dom}+(M+(K-1)^D)\cdot\alpha)$.

5　实验验证与分析

本文提出的算法由 *VC++6.0* 实现,实验在 Windows XP,主频 2.0G 奔腾 4 处理器和 1G 内存的配置上进行. 采用文献[2]提供的合成数据进行实验. 考虑基于计数的滑动窗口数据流模型,滑动窗口的默认宽度(W)设定为 300K,即窗口中包含 300K 个对象. 数据流的流速模拟为 1000 对象/秒. 其它参数默认值:对象维度(D)为 4,概率阈值(p)为 0.3. 实验参数的变化区间:D 为 2～6,W 为 100K～500K,而 p 则分别取 0.1、0.3、0.5、0.7、0.9. 每组实验执行 100 次,结果取平均值. 以下 5.1 节测试网格粒度对 SOPDS 性能的影响,5.2 节测试不同参数下算法的时间性能,5.3 节测试 SOPDS 算法空间剪枝效果. 为更客观地衡量 SOPDS 算法的性能,以此前相关工作为基础了开发一个基准算法 Naive 算法. 具体地,将文献[2]中最高效的算法 P-Lazy 直接扩展到基于计数的概率数据流的环境之下,并对它进行适当的优化. 在 5.2 和 5.3 节中,SOPDS 将与 Naive 进行全面性能对比.

5.1　网格粒度的影响

本节测试 K 值(格索引中每维的等份数)对 *SOPDS* 性能的影响,试图找出使算法性能最优的 K 值. 在本实验中,基本参数配置为:$D=4$, $W=300K$, $p=0.3$. 将 K 值由 1 逐步增长到 8,图 12 给出了在不同 K 值和数据分布下,算法处理每个对象平均花费的时间.



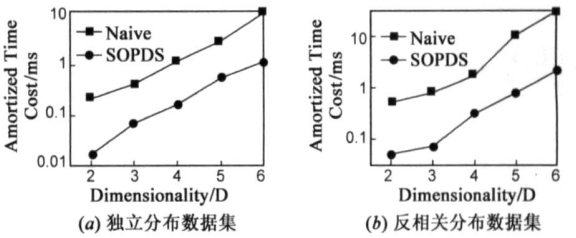
(a) 独立分布数据集　(b) 反相关分布数据集

图12　不同K值下的算法性能

两种分布数据集下算法性能曲线都近似于"U"形. 在其它维度下,性能曲线也类似. 当 K 由 1 增大到 2 时,时间耗费急剧下降. 这是因为 $K=1$ 时,启发式规则 1 与规则 2 均无效. 当 K 为 2 时,启发式规则立即发挥作用. 随着 K 进一步增大,曲线达到最低点后开始反弹. 这是因为过于稀疏的网格将导致单个格中存在较多对象,而过于精细的网格将导致格的数目激增. 综合考虑,维度 2-6 时 SOPDS 所选定的 K 值分别为 8、5、4、4、3,它们将应用于此后的实验中.

5.2　时间开销测试

本节从数据维度、规模与概率阈值三个角度对 SOPDS 与 Naive 的时间性能进行对比. 第一组实验先考察算法的维度可扩展性. 在该实验中 W 和 p 被固定为 300K 和 0.3,将 D 由 2 逐步增加到 6,图 13 给出了实验结果. 当维度由 2 增加到 6 时,在两种分布数据集上,SOPDS 处理一个对象所花费的时间约是 Naive 算法的 1/10-1/15.

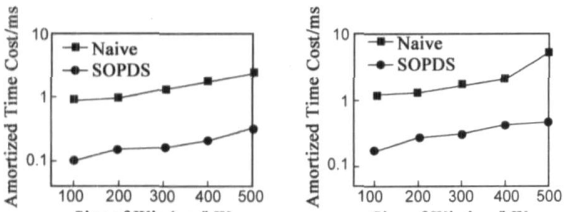


(a) 独立分布数据集　(b) 反相关分布数据集

图13　维度可扩展性实验结果

第二组实验先考察规模可扩展性. 在该实验中 D 和 p 被固定为 4 和 0.3,将滑动窗口的宽度 W 由 100K 逐步增加到 500K,图 14 给出了实验结果. 在独立分布数据集上,随着数据集规模的增大,SOPDS 处理一个对象的平均时间由 0.1ms 增加到 0.32ms,而 Naive 则由

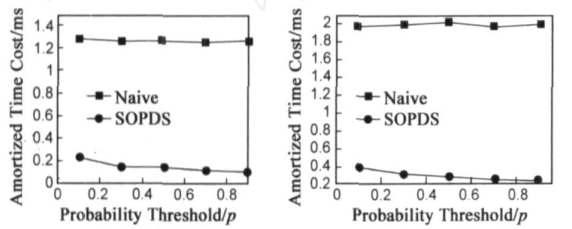
0.89ms 增加到 2.45ms. 在反相关分布数据集上,SOPDS 对 Naive 优势也在一个数量级左右.



(a) 独立分布数据集 (b) 反相关分布数据集

图14 规模可扩展性实验结果

第三组实验先考察算法性能对概率阈值变化的反应. 在该实验中 D 和 W 被固定为 4 和 300 K,将概率阈值由 0.1 逐步增加到 0.9,图 15 给出了实验结果. 随着概率阈值的增大,Skyline 的规模急剧下降;另外一方面,随着阈值的增大,对象更容易达到启发式规则 3 的条件而从系统中淘汰,故系统中保存的候选 Skyline 对象数也会逐渐下降. 这两方面的原因促使 SOPDS 性能逐渐上升. 而在 Naive 算法中,阈值的变化对系统中所



(a) 独立分布数据集 (b) 反相关分布数据集

图15 阈值算法性能的影响

保存的对象数无任何影响,故其性能无明显的变化.

5.3 空间开销测试

本节考察空间优化策略的效果,以 SOPDS 算法中保存的平均对象数与 Naive 算法中保存的对象数之比来衡量. 第一组实验考察维度对剪枝效果的影响. 图 16 给出了实验结果. 随着维度的由 2 增加到 6,在独立分布数据集下,算法的剪枝率由 97.3 %下降至 88.4 %;而在反相关分布数据集下算法的剪枝率由 87.5 %下降至 47.6 %. 第二组实验考察数据规模对剪枝效果的影响. 图 17 给出了实验结果,随着数据规模的增大,剪枝率也缓慢地增大. 这是因为随着数据规模的增大,空间中的对象的密度将逐渐变大,对象达到启发式规则 3 中条件的可能性上升,故剪枝效果愈发明显.

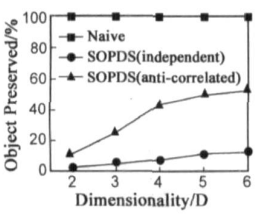


图16 维度的影响

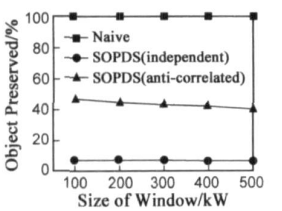


图17 规模的影响

第三组实验考察概率阈值对剪枝率的影响,图 18 给出了实验结果:剪枝率随着阈值同步增大. 这是因为随着阈值的增大,一方面 Skyline 对象数变小,另一方面对象更容易达到启发式规则 3 的条件而从系统中淘汰,故剪枝率升高. 第四组实验考察剪枝率随数据流的演化而波动地情况. 图 19 的实验结果显示系统中保存的对象比例有少许波动,但总地来说比较平稳.

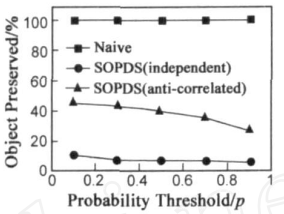


图18 阈值的影响

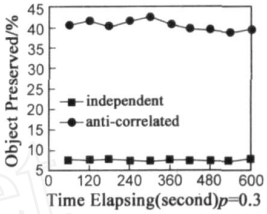


图19 剪枝率波动情况

6 结论

本文对概率数据流上的 Skyline 计算进行了深入地研究,详尽地分析了对象的状态及其相互转换关系,最终提出了一个高效的算法 SOPDS. 在算法的实现技术上,采用网格作为索引,并进一步开发了一系列启发式优化方法使算法具有较高整体性能. 对比实验表明,算法高效并具有良好的可扩展性.

参考文献:

[1] Borzsonyi S, Kossmann D and Stocker K. The Skyline Operator [A]. Proc of ICDE[C]. Washington: IEEE Computer Society, 2001. 421 - 430.

[2] Tao Y, Papadias D. Maintaining sliding window skylines on data streams[J]. IEEE Transactions on Knowledge and Data Engineering (IEEE TKDE), 2006, 18(3): 377 - 391.

[3] Lin X, Yuan Y, Wang W, Lu H. Stabbing the Sky: Efficient Skyline Computation Over Sliding Windows[A]. Proc of ICDE [C]. Washington: IEEE Computer Society, 2005. 502 - 513.

[4] Papadias D, Tao Y, Fu G, Seeger B. Progressive skyline computation in database systems[J]. ACM Transations on Database Systems, 2005, 30(1): 41 - 82.

[5] Chomicki J, Godfrey P, Gryz J and Liang D. Skyline with presorting[A]. Proc of the ICDE[C]. Washington: IEEE Computer Society, 2003. 717 - 719.

[6] Pei J, Jiang B, Lin X, Yuan Y. Probabilistic skylines on uncertain data [A]. Proc of the VLDB [C]. Vienna: ACM Press, 2007. 15 - 16.

[7] Babcock B., Babu S, Datar M, Motwani R, Widom J. Models and issues in data stream systems[A]. Proc of the ACM PODS [C]. New York: ACM Press, 2002. 1 - 16

[8] Cormode G, Garofalakis M. Sketching probabilistic data streams [A]. Proc of the ACM SIGMOD [C]. Beijing: ACM Press,

2007. 281 - 292

[9] Dalvi N, Suciu D. Efficient query evaluation on probabilistic databases[A]. Proc of VLDB[C]. Toronto: Morgan Kaufmann Publishers, 2004. 864 - 875.

[10] Burdick D, Deshpande PM, Jayram TS, Ramakrishnan R, Vaithyanathan S. OLAP over uncertain and imprecise data [A]. Proc of VLDB[C]. Trondheim: ACM Press, 2005. 970 - 981.

[11] Sarma AD, Benjelloum O, Halevy A, Widom J. Working models for uncertain data [A]. Proc of ICDE[C]. Washington: IEEE Computer Society, 2006.

[12] Cheng R, Kalashnikov D, Prabhakar S. Querying imprecise data in moving object environments[J]. IEEE Trans on Knowledge and Data Engineering, 2004, 16(9): 1112 - 1127.

[13] Ngai WK, Kao B, Chui CK, Cheng R, Chau M, Yip KY. Efficient clustering of uncertain data [A]. Proc of the ICDM [C]. Hong Kong, 2006. 436 - 445.

[14] Jayram TS, McGregor A, Muthukrishan, Vee E. Estimating statistical aggregates on probabilistic data streams[A]. Proc of the ACM PODS[C]. Beijing: ACM Press, 2007. 243 - 252.

[15] Jayram TS, Kale S, Vee E. Efficient aggregation algorithms for probabilistic Data[A]. Proc of the ACM-SIAM SODA[C]. Louisiana, 2007. 346 - 355.

[16] 孙圣力,黄震华,李金玖,郭建奎,朱扬勇. 数据流上高效计算子空间 Skyline 的算法[J]. 计算机学报, 2007, 30(8), 1418 - 1428.
Sun S L, Huang Z H, Li J J, Guo J K, Zhu Y Y. Efficient computation of subspace skylines over data streams. Chinese Journal of Computers, 2007, 30(8), 1418 - 1428. (in Chinese)

[17] 刘旭,毛国君,孙岳,刘椿年. 数据流中频繁闭项集的近似挖掘算法[J]. 电子学报, 2007, 35(5), 900 - 905.
Liu X, Mao G J, Sun Y, Liu C N. An algorithm to approximately mine frequent closed itemsets from data streams[J], ACTA Electronica Sinica, 2007, 35(5), 900 - 905. (in Chinese)

作者简介:

孙圣力　男, 1979 年生于湖南省澧县, 博士, 北京大学软件与微电子学院讲师, CCF 会员. 主要研究方向为数据库与数据仓库、在线分析处理、数据挖掘.
E-mail: slsun@ss.pku.edu.cn

戴东波　男, 1977 年生于湖南省娄底, 复旦大学计算机科学技术学院博士研究生. 主要研究方向为流数据管理与挖掘, 生物信息学.