

基于网格和队列触发的多维空间 Skyline 查询算法

张 斌, 孟凡荣, 闫秋艳

(中国矿业大学 计算机科学与技术学院, 江苏 徐州 221116)

摘 要: 在流数据环境下, 多维空间的 Skyline 计算会随着维度的增高影响滑动窗口上 Skyline 点集的计算效率. 针对以上问题, 提出 SQTS 算法, 采用网格索引结构, 对元组进行标记, 排除部分非 Skyline 点集; 引入 SD 队列触发 Skyline 维护, 避免过期数据点的频繁排查, 提高多维空间下的 Skyline 计算效率. 仿真实验结果表明, SQTS 算法在多维流数据环境下, 提高了 Skyline 点集的计算效率, 验证了算法有效性.

关键词: 数据流; 滑动窗口; 多维空间; Skyline; 网格

中图分类号: TP31

文献标识码: A

文章编号: 1000-7180(2010)08-0108-04

Grid and Queues Triggering Based Algorithm for Hyperspace Skyline Queries

ZHANG Bin, MENG Fan-rong, YAN Qiu-yan

(School of Computer Science and Technology, China University of Mining and Technology, Xuzhou 221116, China)

Abstract: Under the environment of data stream, Skyline calculation of the multi-dimensional space will affects the computational efficiency of Skyline point sets of in the sliding windows with the increase of dimension. In order to solve this problem, this paper proposes SQTS algorithm, which tagging tuples by grid index structure to exclude the point sets that not belong to Skyline. Besides, SQTS algorithm introduces SD queues to trigger Skyline maintenance, which avoids checking the overdue data points frequently and improves the calculation efficiency of Skyline of multidimensional space. Finally, this paper gives the results of simulation experiments, which demonstrates that under the environment of multidimensional space, SQTS algorithm improves the calculation efficiency of Skyline point sets; that is to say, SQTS algorithm in this paper is available.

Key words: data stream; over sliding windows; hyperspace; Skyline; grid

1 引言

数据流是连续的、随时间变化的、无界的数据项序列^[1]. Skyline 查询也称为 Pareto 查询, 是一个典型的多目标优化的问题^[2-3]. 在数据流模型中, 单个的数据项通常采用关系元组的形式, 经过处理后就会被销毁, 所以数据流算法是建立在一次查询的基础上的. 因此, 在数据流上的 Skyline 查询计算, 也是基于一次查询基础上的, 更注重实时的查询效率.

2005 年, Lin 等人在 ICDE 会议上提出了第一

个数据流上的 skyline 查询处理算法^[4], 该算法基于 R-tree 索引结构, 计算大小为 N 的滑动窗口轮廓; 2006 年, Morse 等人在 ICDE 会议上提出了一种新的 skyline 查询处理算法 LookOut 算法^[5], 该算法第一次考虑到了生存周期; 同年, Tao^[6]提出的 I-Eager 算法, 基于 R-tree 结构来组织, 用 d -向范围搜索来支持后续操作. 但在构造 R-tree 的时候, 会产生较多的 MBR(最小边界矩形)的相互重叠, 而且在维度稍高的时候搜索效率会急剧下降^[7]. 文献[8]提出的基于网格的 Skyline 查询算法 GUSM, 由于滑动

收稿日期: 2010-06-08; 修回日期: 2010-07-08

基金项目: 国家自然科学基金项目(50674086); 江苏省博士后科学基金项目(0701045B); 中国矿业大学科技基金项目(2007B017)

窗口中的过期点会导致 Skyline 点集频繁更改,因此需要频繁对过期点排查,而且随着维度升高,计算消耗时间涨幅较快。

本文基于 Tao^[6]的 I-Eager 和文献[8]的 GUSM 算法,提出多维流数据环境下的 Skyline 计算方法, SQTS(Skyline Queue Trigger of Skyline). 该算法采用网格索引结构,由同步队列触发 Skyline 查询更新,解决了由于维度增高影响滑动窗口上 Skyline 点集的计算效率问题,仿真实验结果表明, SQTS 算法在多维流数据环境下可以有效提高 skyline 点集的计算效率,验证了算法的有效性。

2 问题描述

2.1 问题定义

假设有 d 维的空间 $D, D = \{a_1, a_2, \dots, a_d\}$, 其中属性 $a_1, a_2, \dots, a_d$ 的定义域均为区间 $(0, 1]$, 本文提到的所有元组均来自空间 D 。

定义 1 (支配^[9]) 假定数据点集 $p = (p_1, p_2, \dots, p_n), P \subseteq D$, 当且仅当点 p_i 在任意维度上的值小于 p_j , 我们就说点 p_i 支配点 p_j . 我们记作 $p_i \rightarrow p_j$.

定义 2 (Skyline 集合^[2]) 在 d 维空间 D 的数据点集中, Skyline 集合是由那些在所有维上都不被任何其他点支配的点组成的集合。

本文常用的符号及意义, 如表 1 所示。

表 1 常用符号及意义

符号	意义	符号	意义
W	滑动窗口的大小	SD	Skyline 队列
d	维度	PL	指针列表
DB	系统中目前存在的元组	CL	影响区域指针
DB_{sky}	一组正存在的 Skyline 点	$r.t_{arr}$	元组 r 的到达时间
DB_{rest}	DB 中非 Skyline 元组	$r.t_{exp}$	元组 r 的终止时间

在流数据环境下, Skyline 集合也会随着元组的到达和销毁, 做比较频繁的改变. 所以, 必须对元组进行标记, 以便对其进行操作. 因此, 将数据流中对象到达系统的时间戳也看作对象的一个属性, 时间戳为整数, 自 0 开始递增. 即任意一个对象 r 可以记作 $r(id, r.t_{arr}, r.val_1 \dots r.val_n)$, 其中 $r.t_{arr}$ 就表示其到达系统的时间戳, $r.val_1$ 表示在属性 a_1 上的取值. 本文是基于滑动窗口模型的, 设滑动窗口大小为 W , 则对任意一个对象仅在时间区间 $[r.t_{arr}, \dots, r.t_{arr} + W - 1]$ 上是有效的. 因此, 只需考虑窗口中存放的最近到来的 W 个元组中的 SK.

2.2 数据集变化对 Skyline 造成的影响

根据 2.1 节可知, 流数据的元组会不断的到达

系统, 到期销毁. 因此, 结合 Skyline 集合的定义, 以及数据流特点, 参照文献[4-6]给出以下定理, 以便对数据流 Skyline 查询算法进行深入研究。

定理 1 新增点 r 时:

(1) 若 $\exists e \in DB_{sky}$ 且 $e \rightarrow r$, 则 DB_{sky} 不发生变化;

(2) 若 $\forall e \in DB_{sky}$ 且 $e \rightarrow r$ 均不成立, 则 $DB_{sky} = DB_{sky} + \{r\}$;

(3) 若 $\exists e \in DB_{sky}$ 且 $r \rightarrow e$, 则 $DB_{sky} = DB_{sky} + \{r\} - S$, 其中 S 为所有 e 的集合。

定理 2 点 r 失效时:

(4) 对 $\forall e \neq r$, 若 r 失效前 $e \in DB_{sky}$, 则失效后, $e \in DB_{sky}$ 依旧成立;

(5) 若 $r \in DB_{rest}$, 则 DB_{sky} 不发生变化;

(6) 若 $r \in DB_{sky}$, 则 $DB_{sky} = DB_{sky} - \{r\} + S$, 其中 S 为仅被 r 唯一支配的集合。

3 SQTS 算法

3.1 算法思想

本文提出的 SQTS 算法, 采用网格索引数据结构提高效率, 系统中所有相近的点用规则的网格统一进行管理, 对于系统中的点集, 则可以用其所在的网格表示, 根据所在的网格, 就可进行初步比对, 提前排除掉部分非 Skyline 点集, 提高比对效率; 同时引入 Skyline 队列 SD 触发 Skyline 维护, 避免排查因数据点过期对 Skyline 点集造成的频繁更改工作, 提高计算效率。

3.2 SD 队列

在数据流环境下对 Skyline 维护, 需要对过期数据进行排查, 检索过期 Skyline 点; 而当某 Skyline 点过期时, 需要重新计算 Skyline 点集. 本文提出 Skyline 队列 SD, SD 在时间轴上与滑动窗口同步. 在每次滑动窗口进行下一跳时, 不再对到期的数据做任何排查而直接丢弃. 而当 SD 推出某过期点, 即为原 Skyline 中过期点, 而在该点指向的集合中排查出的 Skyline 集合就是过期点的替补, 因此, 避免了对当前 Skyline 的排查, 缩小替换过期点时的排查范围, 提高了维护效率。

3.3 基于网格的 SQTS 算法

不失一般性, 以二维空间为例. 如图 1 所示. 每一个对象 r , 当前状态表示为 $(id, r.t_{arr}, x_1, x_2)$, 而整个空间被划分成若干个等宽的网格. 不妨设网格任意一维上的宽度为 δ , 那么, 任意一个网格 $C_{i,j}$ 就包含了 x_1 区间在 $(i \cdot \delta, (i+1) \cdot \delta]$ 且 x_2 区间在 $(j$

$\cdot \delta, (j+1) \cdot \delta]$ 上的所有点;反之,对于任何一个点 r , 可以算出它所在的网格 $C_{i,j}$, 其中 $i = \lfloor x_1/\delta \rfloor, j = \lfloor x_2/\delta \rfloor$. 用 $C_{i,j} \cdot BL$ 和 $C_{i,j} \cdot TR$, 分别表示 $C_{i,j}$ 的左下和右上的顶点. 则显然对于 $C_{i,j}$ 中包含的任意点 e 都有 $C_{i,j} \cdot BL \rightarrow e$, 和 $e \rightarrow C_{i,j} \cdot TR$.

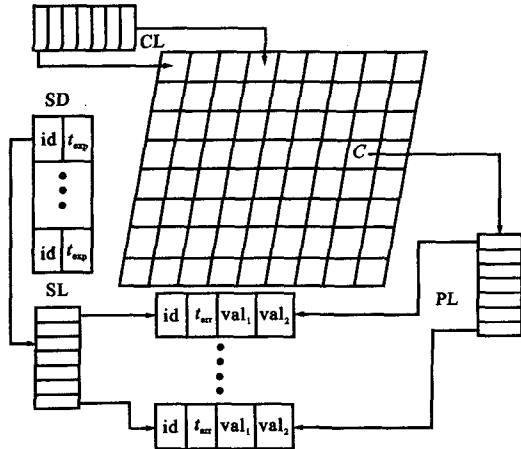


图1 网格索引数据结构

每一个网格包含一个指针列表 PL, 指向它所包含的点; 系统还包含一个 Skyline 的指针列表 SL, 指向目前系统中所有的 Skyline 点; 一个网格影响区域指针 CL, 来近似表示影响区域, 以及一个 Skyline 维护触发队列 SD, 里面按照 Skyline 点的失效时间顺序存储, 并且每个点都有一个指针指向自己唯一支配的集合中, 来触发 Skyline 的维护。

假设一个点 e 属于 SK, 如果 $e \rightarrow C_{i,j} \cdot BL$, 那么我们称为完全支配, 记作 $e \rightarrow_a C_{i,j}$; 如果 $e \rightarrow C_{i,j} \cdot TR$, 那么我们成为不完全支配, 记作 $e \rightarrow_u C_{i,j}$, 对于完全支配的, 我们可以直接排除掉落在 $C_{i,j}$ 中的所有点集, 对于不完全支配的, 我们需要进行进一步比对, 确定排除点。

SQTS 算法主要由 SD 队列触发 Skyline 维护, 以及新进点排查两方面构成, 具体功能, 见 3.4 节算法描述。

3.4 算法描述

给出触发维护以及新进点排查的算法描述。

以下是触发的算法描述:

算法: 触发算法

输入: 数据流元组

输出: 新 Skyline 点集

过程:

- (1) SD 推出一个 Skyline 点 e ;
- (2) 删除相对应 SL 中的 e 的指针, 并从 DB_{sky} 中删除;
- (3) if(e 唯一支配的集合 S , 未到期)
- (3.1) S 加入 DB_{sky} , 推入 SD, 标注到期时间 t_{exp} 并依次

排序, S 指针加入 SL;

(4) else

(4.1) 排查 e 临近点选出新 Skyline 点

(5) 将新点加入 DB_{sky} , 推入 SD, 标注到期时间 t_{exp} 并依次排序, 指针加入 SL;

以下是新进点排查的算法描述:

算法: 新进点排查算法

输入: 数据流元组集合 S

输出: 新 Skyline 点集

过程:

- (1) 新入滑动窗口点集合 S
- (2) if(存在 Skyline 点集 $\rightarrow$ 新进点集合的网格 $C_{i,j}$)
- (2.1) 新近点存入 DB_{rest}
- (3) else if(存在 Skyline 点集 $\rightarrow_u$ 新进点集合的网格 $C_{i,j}$)
- (3.1) 将新进点集合与不完全支配它的点集共同排查出 Skyline 点集合 S
- (4) S 加入 DB_{sky} , 推入 SD, 标注到期时间 t_{exp} 并依次排序, S 指针加入 SL;
- (5) 其他点集存入 DB_{rest}

4 实验分析

4.1 实验数据说明

实验采用模拟数据, 对各元组每维属性分配随机数据, 共有 1 000 000 个数据点, 持续到达来模拟数据流场景. 窗口宽度为 10 000, 即窗口保留 10 000 个数据点, 流速 100 元组/秒. 实验对比了 I-Eager, GUSM, SQTS 三种算法随着维度升高的时间消耗。

实验平台:

Memory: 1G CPU: Inter(R) Pentium(R) D
CPU 1.60GHz

OS: Microsoft Windows XP Professional Service
Pack 2

PL: Microsoft Visual C++ 6.0

4.2 实验及分析

由 4.1 节可知, 每维网格数目对算法的影响会很大. 网格越小, 即每维网格数目越多, 则网格的近似影响区域就更准确, 降低运算规模, 但随着网格数目的增多, 会增大整个网格的构建、处理开销; 反之, 网格越大, 每维网格数目越少, 则网格的近似影响区域就比较模糊, 增加了运算规模. 以五维数据为例, 如图 2 所示, 随着每一维度网格数目的增加, 时间消耗先减小后增大, 在每一维度划分为 20 个网格时, 时间消耗最低. 为便于对比平均运算时间, 相关测试均建立在每维网格划分为 20 个的基础上进行。

图 3 为试验中对比了 SQTS, I-Eager, 以及

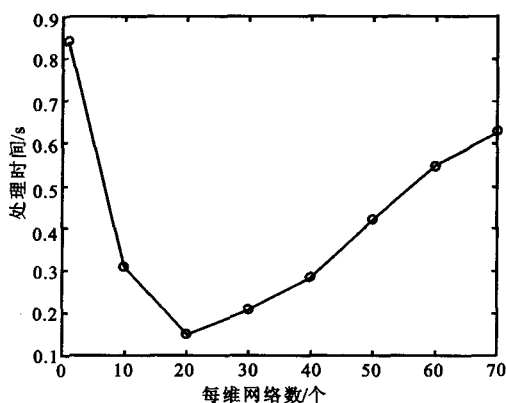


图2 每维网格数与处理时间关系图

GUSM,三者都基于相同的数据源,而 GUSM 和 SQTS 每维具有相同的网格数。由图易知, I-Eager 基本上趋于一个线性增长,且在低维时,处理时间高于另外两种算法 10~15 倍。在维度较低的时候($d < 3$)的时候, SQTS 与 GUSM 二者平均运算时间消耗差距不大;而随着维度的增高,两者的时间消耗都成非线性的增长。但 SQTS 的增长幅度要小于 GUSM。根据 3.1 节分析可知,在维度增高以后,必然会导致 Skyline 的维护工作以及搜索效率急剧下降,在引入 SD 队列之后,在一定程度上减少了对 Skyline 维护工作的次数,从而较好的解决这个问题,很大程度上减少了计算量,没有造成过多的时间消耗。但与此同时,根据 3.2 节描述可知,随着维度的增加,大量的存储开销会随之呈指数级增长。从数据流的性质中,我们可知,在数据流环境下,我们更关注处理效率问题,在存储开销允许的情况下,处理效率优先。

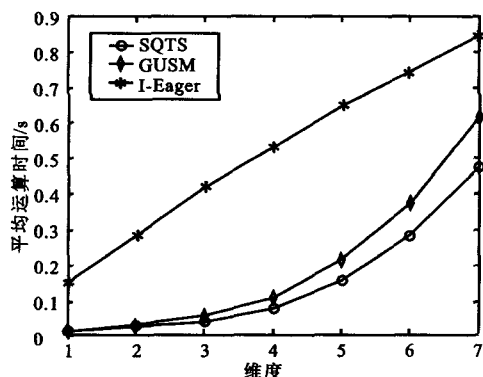


图3 处理时间随数据维度变化图

5 结束语

本文受到 I-Eager 和 GUSM 算法的启发,在数据流环境下,引入网格作为索引结构,提出了带有触

发性维护的 SQTS 算法,提高多维空间下的 Skyline 计算效率。实验对比分析了三种算法随维度增加对时间效率的影响,证明了在维度增加时, SQTS 算法本身的效率得到很大程度的提高。

在本文中,采用的是网格结构来提高效率问题,但随着维度增加,会导致大量存储开销,怎样进一步改进结构组织,将是下一步工作的重点。

参考文献:

- [1] Babcock B, Babu S, Datar Metal. Models and issue in data stream systems[C]// Proceedings of 21th ACM SIGMOD - SIGACT - SIGART Symposium on Principles of Database Systems. Madison, 2002: 1-16.
- [2] Borzsonyi S, Kossmann D, Stocker K. The skyline operator [C]// Proceedings of the 17th Int'l Conf. on Data Engineering. Heidelberg, IEEE Computer Society Press, 2001: 421-430.
- [3] Mitchell T M. Machine learning[M]. New York: McGraw - Hill Companies Inc, 1997: 230-247.
- [4] Lin X, Yuan Y, Wang W. Stabbing the sky: efficient skyline computation over sliding windows[C]// Proceedings of the 21st International Conference on Data Engineering (ICDE 2005). Tokyo, Japan, 2005: 502-513.
- [5] Morse M, Patel J M, Grosky W I. Efficient continuous skyline computation[C]// Proceedings of the 22nd International Conference on Data Engineering (ICDE 2006). Atlanta, USA, 2006: 108.
- [6] Tao Y, Papadias D. Maintaining sliding window Skylines on data streams[J]. IEEE Trans on KDE, 2006, 18(2): 377-391.
- [7] Theodoridis Y, Sellis T. A model for the prediction of R-tree performance[C]// Proceedings of the ACM PODS. Montreal, Canada, 1996: 161-171.
- [8] 李田. 更新数据流上的连续 Skyline 计算[J]. 计算机工程与科学, 2008, 30(5): 59-83.
- [9] Guttman A. R-tree a dynamic index structure for spatial searching [C]// Proceedings of the ACM SIGMOD. Boston, Massachusetts, USA, 1984: 47-57.

作者简介:

张斌男, (1986-), 硕士研究生. 研究方向为数据挖掘和知识发现。

孟凡荣女, (1962-), 教授, 博士生导师. 研究方向为数据库技术和知识发现。

闫秋艳女, (1978-), 博士研究生. 研究方向为流数据处理、时间序列处理。