

多维数据流下的 top-k 支配查询算法研究

陈加付, 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: Top-k 支配查询算法返回前 k 个支配得分最高的数据对象, 在多维数据查询中有效地限定了结果集的大小, 但是无法直接应用于数据流查询中. 文中使用网格索引结构在多维数据流下构建的 top-k 支配算法, 采取动态创建网格, 反向遍历一次求支配区点数, 剪枝无效单元格等策略, 提高了查询的效率. 实验结果证明了该方法在多维数据流查询中的有效性.

关键词: 数据流; 滑动窗口; top-k; Skyline; 网格

中图分类号: TP311

文献标识码: A

文章编号: 1000-7180(2010)08-0218-04

Study on Top-k Dominating Query Algorithm over Multi-dimensional Data Streams

CHEN Jia-fu, MENG Fan-rong, YAN Qiu-yan

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

Abstract: The top-k dominating query returns k data objects which dominate the highest number of objects, which effectively limits the result size in multi-dimensional data query. However, it cannot be applied to data streams query directly. Using the grid index structure, this paper builds a top-k dominating query model over multi-dimensional data streams. The model improves the query efficiency with the strategy of dynamic creation of grid, traversing the grid to get the points number in dominated region in reversed order and pruning invalid cells. Experiment results prove the validity of the method in multi-dimensional data stream query.

Key words: data streams; sliding window; top-k; Skyline; grid

1 引言

数据流具有实时、无限等特征, 是一种动态时变数据^[1]. 特别是多维数据流查询, 一直是近年来的研究热点. top-k 算法进行多维数据流查询时使用人为定义的分级函数^[2], 分级函数的选取直接决定了结果集, 这种选取的随意性, 无法保证结果集的客观性. skyline 在多目标决策中有着重要作用^[3], 但是算法在多维数据流查询中^[4], 往往又会产生一个无法限定大小的结果集, 这样即使获得了 skyline 集, 因为数据量的庞大, 也无助于对现实问题进行分析. 为解决这个问题, 文献[5]提出了 top-k 支配查

询, 该算法去除了 top-k 中的评价函数, 限定了结果集的大小, 同时不会像单纯的进行 skyline 查询那样, 遗漏那些潜在的, 非常重要的点^[5], 但是该算法是基于 R-树结构的, 算法的效率难以满足在数据流这种快速更新数据的情况下的查询需求.

文中采用文献[2]中所述的网格索引结构, 对文献[5]中的算法进行改进, 提出了 GTS (Based on grid top-k & skyline query algorithm) 算法, 并且对 GTS 采用了一些优化策略, 构建了 OGTS (Optimized GTS) 算法, 进一步提高其在多维数据流查询中的效率.

2 top-k 支配查询相关概念

文献[5]中给出了 top-k 支配查询的相关的概念:假设一个数据集 D , 它所包含的点有 n 维空间 R^n , 则数据集表示为 $D(d_1, d_2, \dots, d_n)$.

定义 1 给出一个单调的分级函数 F , 假定小值比大值更好, 那么一个 top-k 查询返回数据集 D 中前 k 个 F 值最小的点.

定义 2 假定小值比大值更好, 则 p 支配 p' , 有 $(\exists i \in [1, d], p[i] < p'[i]) \wedge (\forall i \in [1, d], p[i] \leq p'[i])$ ($p[i]$ 标注了点 p 在 i 维度上的值.) p 支配 p' 数学表示为: $p > p'$.

定义 3 一个 skyline 查询返回数据集 D 中所有不被其它点支配的点.

一个支配函数 $\tau(p)$ 表示被点 p 支配的点数, $\tau(p) = |\{p' \in D \mid p > p'\}|$, 为便于描述中称之为支配因子.

定义 4 top-k 支配查询: 一个 top-k 支配查询返回前 k 个 τ 值最高的点.

文中发现了一个关于支配因子的推论: 存在两个点 p_1, p_2 , 如果 $\tau(p_1) \leq \tau(p_2)$, 那么 p_1 必定不能支配 p_2 .

该推论可以通过反证法证明: 假设 $p_1 > p_2$, 那么根据支配的定义, p_1 支配的点数至少为 $\tau(p_2) + 1$, 这与 $\tau(p_1) \leq \tau(p_2)$ 矛盾, 则该推论成立, 进一步证明了 top-k 支配查询算法的客观性.

3 基于网格的 top-k 支配查询算法

3.1 网格结构

数据流数据的频繁更新使得文献[5]中的 R 树结构不再适合使用在数据流查询场景中, 文献[2]中的网格结构却能很好的满足这一需求. 这个部分将描述本文算法采用网格结构的合理性. 不失一般性, 假定一个二维空间 $D(d_1, d_2)$ 中, 每一个数据点描述为 $p(id, x, y)$, 其中 id 为每一个点的唯一标志; x, y 分别对应 d_1, d_2 维的数值. 文中采用平均分割网格的方法来划分网格, 即在各自维度上的划分刻度都是相等的. 如图 1 中, 在 d_1 维度上, 刻度为 a ; 在 d_2 维度上, 刻度为 b ; d_1, d_2 维上的网格划分数 $div = 4$, 这样在几何上就会直观地形成 4×4 的网格. 一个新到达的点 $p_{new}(id, x, y)$ 将会被放置在单元格 $C_{x/a, y/b}$ 中. 通过这种方式, 可以将一个新到达的点迅速地定位到它所在的单元格 C 中.

网格结构下, 一个相关概念: 一个单元格

$C(d_1, d_2, \dots, d_n)$ 的相邻单元格是指: 在 d_1 到 d_n 维度上有且仅有一个维度上的下标是与 C 相等的, 其他维度上都大于 C 中相应维度下标的单元格.

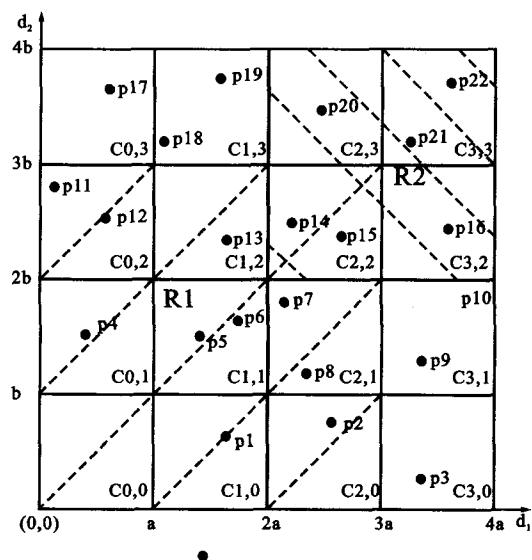


图 1 在 4×4 网格中点的分布图

在图 1 中, 假设计算一个落在 $C_{1,1}$ 中的一个点 $p = p_5$ 的支配的所有点数 $\tau_s(p)$ 时, 会分为 3 种情况进行计算:

(1) 统计 $C_{1,1}$ 的支配区域 R_2 中所有的点, 区域 R_2 中的点数为 $m = \text{count}(R_2) = 6$;

(2) 进行 skyline 计算, 获得与 $C_{1,1}$ 的相邻单元格 $C_{1,2}, C_{1,3}, C_{2,1}, C_{3,1}$ 中点 p 可以支配的点数.

(3) 在单元格 $C_{1,1}$ 自身中 p 可以支配的点数记为 $\tau(p)$, 那么计算点 p 的支配因子之和 $\tau_s(p)$ 为

$$\tau_s(p) = m + n + \tau(p) = 11$$

在这个计算过程中, 无需对那些被 $C_{1,1}$ 支配的区域 R_2 中的点进行支配计算, 可以看出在采用网格索引结构完成 top-k 支配计算时, 可以极大的减少计算量.

3.2 OGTS 算法构建

GTS 算法的构建如图 2 所示. 将滑动窗口中的数据以网格索引结构来重新组织, 整个网格通过哈希表结构来实现, 网格哈希表中每一个成员的组织形式是 $(C_{id}, *p_c, \tau_c, \tau_r)$, C_{id} 为该单元格的唯一标志 id , $*p_c$ 为指向该单元格哈希表的指针, τ_c 标明了该单元格的包含的点数, τ_r 标明了该单元格支配区内包含的点数. 单元格哈希表中每一个成员的组织形式是 $p(id, d_1, d_2, \dots, d_n)$. 因为哈希表在查找定位点时花费的时间为常数时间, 所以通过这种数据组织方式可以快速的完成查找, 插入和删除一个点, 而且在计算 3.1 节中的 $m = \text{count}(R_2)$ 时, 只需

要获得该单元格所支配区域网格哈希表中成员的 τ_c 之和即可, 无需访问具体的单元格哈希表, 节省了查找时间。

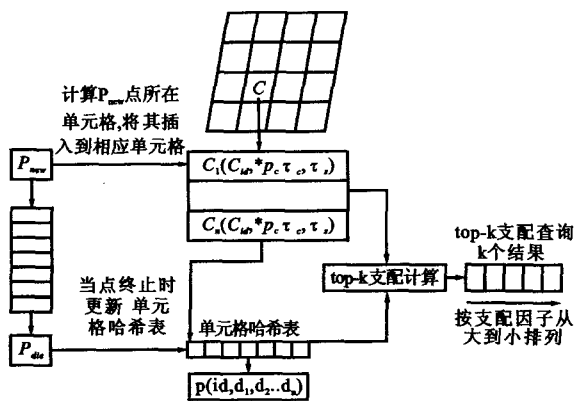


图2 基于网格的 top-k 支配算法框架

GTS 算法的流程:

(1) 对每一个新到达的点, 计算其所应在的单元格, 将其插入到相应单元格中, 更新该点所在单元格的 $\tau_c = \tau_c + 1$ 。

(2) 对于每一个终止时间到达的点, 根据该点的 id 将该点从所在单元格哈希表中删除, 更新该点所在单元格的 $\tau_c = \tau_c - 1$ 。

(3) 遍历网格, 为每一个点 p 计算 $\tau_s(p)$, 按 $\tau_s(p)$ 取前 k 个结果, 将前 k 个结果按 $\tau_s(p)$ 从大到小的顺序插入到 top-k. List 链表中, top-k. List 链表中数据成员形式为 $p(id, \tau_s(p))$ 。

文中为了进一步提高 GTS 算法的效率, 采取了下列优化策略。

(1) 在构建网格时, 采取动态创建单元格的方式。即一个单元格中没有数据时, 将不会创建该单元格, 或者单元格中的所有数据终止时间到达时, 将销毁该单元格。这样算法可以大大减少创建单元格的数量, 减少空间消耗和访问单元格的次数。

(2) 在遍历网格获得 top-k 支配结果集时, 按单元格下标之和 S 从小到大的遍历顺序进行, 由图 1 可以看出这种方式可以实现逐层访问网格, 而且保证了访问网格的顺序是从最有可能成为最终结果集的单元格开始。在实验过程中, 发现如果正常遍历网格时, 在求 3.1 节的第一个步骤, 求支配区点数之和时, 每查询一个单元格时都需要遍历一次该单元格支配区单元格的 τ_c , 这样大量的时间花费在遍历上, 本文采取反向遍历一次网格的方法, 即按照网格的下标之和从大到小遍历网格一次, 即可获得每一个单元格的支配区域的 τ_r 。如图 1 中, 在两维网格

中, 为单元格分别在维度 d_1 和 d_2 增加一个标志该单元格相邻单元格包含点数的属性 a, b 则每一个单元格为 $(C_{id}, p_c, \tau_c, \tau_r, a, b)$, 查看一个单元格 $C_{d1, d2}$ 时, 对该单元格下标加 1, 则

$$C_{d1, d2} \cdot a = C_{d1+1, d2} \cdot \tau_c + C_{d1+1, d2} \cdot a \quad (1)$$

$$C_{d1, d2} \cdot b = C_{d1, d2+1} \cdot \tau_c + C_{d1, d2+1} \cdot b \quad (2)$$

$$C_{d1, d2} \cdot \tau_r = C_{d1+1, d2+1} \cdot \tau_c + C_{d1+1, d2+1} \cdot a + C_{d1+1, d2+1} \cdot b \quad (3)$$

通过以上计算方式, 可以快速稳定的获得每一个单元格的 τ_r , 极大的减少了遍历网格次数。

(3) 剪枝策略: 从图 1 中容易看到每个单元格的左下角 $C_L(d_1, d_2, \dots, d_n)$, 对于该单元格, 该点是最好的 skyline 点, 该点支配的点数记为 $\tau_s(C_L)$, 设定 top-k. List 中第 k 个元组支配的点数为 top_kScore , 如果 $\tau_s(C_L) < top_kScore$, 则忽略该单元格中的点, 而且不再检查被该单元格支配的所有单元格。

基于以上优化策略构建 OGTS 算法, 并给出具体的描述:

输入: 新到达的元组

输出: 前 k 个 τ 值最高的点

令 $p_{arr} = \text{set of } p_{new}$, $p_{off} = \text{set of } p_{die}$

对每一次循环处理, 做下列操作:

(1) For each p_{new} in p_{arr}

(2) 将 p_{new} 插入到相应单元格, 令 $\tau_c = \tau_c + 1$;

(3) For each p_{die} in p_{off}

(4) 将 p_{die} 从相应单元格中删除, 令 $\tau_c = \tau_c - 1$;

(5) 根据单元格下标和 S 从大到小反向遍历网格获得每一个单元格的 τ_r ;

(6) 根据单元格下标和 S 从小到大逐层访问单元格;

(7) for each 单元格 C

(8) if(该单元格没有被标志的单元格支配)

(9) if($C_L \cdot \tau_s > top_kScore$)

(10) for each 点 p in 单元格 C

(11) if($p \cdot \tau_s > top_kScore$)

(12) 将点 p 按 $p \cdot \tau_s$ 从大到小插入到 top-k. List;

(13) else

(14) 标志该单元格, 访问下一个单元格;

(15) 返回 top-k. List 中的结果集。

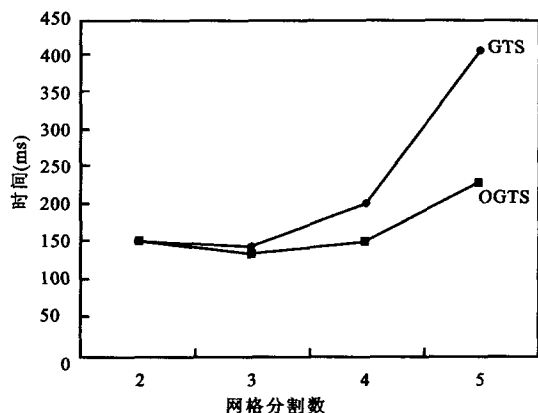
4 实验及分析

本次实验环境为: 惠普 pc 机, CPU: Inter(R) Pentium(R)D 1.60GHz 内存: 1GB, 操作系统: Windows XP Professional. 用 C++ 编程语言实现算法。数据采用在每一个维度上随机独立生成的数据作为

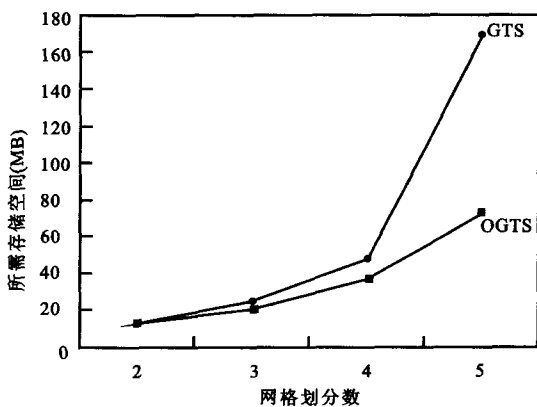
本次实验的测试数据。

实验对 GTS 与 OGTS 之间时间和空间的消耗进行比较,并且将 OGTS 与文献[2]提出的 SMA 算法的时间效率进行了一个对比,以下实验的结果集值均取前个点。

图 3 的实验结果显示了 GTS 和 OGTS 的时间和空间的效率。实验图 3(a)是在维度 $d=8$ 的情况下,元组到达速率 $r=1000$ 个每秒的情况下进行。图 3(a)显示这两者算法在从网格分割数从 2 逐步增加的过程中,消耗时间先是逐步减少,而后逐步增加。GTS 和 OGTS 都是在 $\text{div}=3$ 时,消耗时间达到最少。图 3(b)则是在滑动窗口中的元组数目达到 10000 个时两个算法各自在空间上的消耗。在同维度下,随着每一维度网格分割数的增加,这两者的空间消耗都在逐步增加,但是两者的增长速率却有较大的差异。



(a) $d=8, r=1000$ 处理时间



(b) $d=8, N=10000$ 空间消耗

图 3 GTS 和 OGTS 的时间和空间效率

分析:当网格分割数从 2 开始逐步向上增加时,网格中的单元格呈现的指数型增长,所以在图 3(b)中看到没有采用动态创建网格方法的 GTS 算法空间消耗的增长速度非常快。随着网格分割数的增加,一方面导致单元格本身内的点和相邻单元格内的点

减少,而支配区域内的点相应增多,减少了单元格内和相邻单元格内支配计算的次数;另一方面,随着单元格数的增加,这导致在索引单元格时花费时间增多,必然导致算法效率的下降,所以算法的效率会随着的增加一开始有所提高,随后又会逐步降低。由于 OGTS 采取了优化策略,不会产生空的单元格,有效减少了单元格数目,索引效率得到提高。所以在时间和空间上的效率都比 GTS 好,实验结果和分析保持了一致性。

如图 4,实验是在数据维度 $d=8$, 网格分割数 $\text{div}=3$, $r=100$ 个每秒逐步增加到 1000 个每秒的情况下进行。观察到随着 r 的增加,OGTS 的时间花费相对于 SMA 增长相对缓慢,在 r 较高的情况下 OGTS 有着更好的处理效率,但在 r 较低的情况下, SMA 的算法效率要好于 OGTS,这是因为滑动窗口每跳一次,OGTS 需要重新计算结果集,而 SMA 只是对新到达的点或过期的点进行处理;但是随着 r 的增加,由于在网格结构下将元组分担到各个单元格中,这时 OGTS 的剪枝策略发挥了较好的作用,使得在 r 增加的情况下,OGTS 的时间消耗的增长幅度比 SMA 低,也使得 OGTS 在 r 较高的情况下有着更好的处理效率。

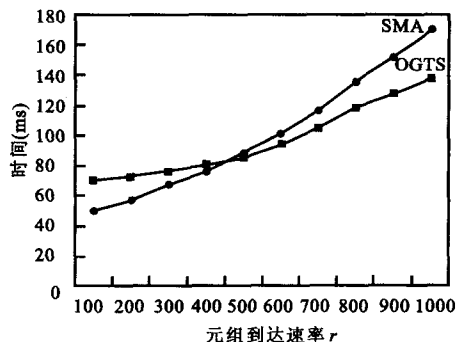


图 4 在滑动窗口下时间效率的对比 ($d=8, \text{div}=4$)

5 结束语

研究结果表明网格结构和文中提出的优化策略的应用,使得 top-k 支配查询可以有效地应用在高维数据流查询中。但是在实验过程中,发现由于 OGTS 算法采用平均分割网格的方法会导致部分网格内数据点过多或者过少,这影响了单元格和相邻单元格内的支配计算效率。为解决平均分割网格而导致的问题,还需要对如何动态分割网格做进一步的研究。

(下转第 225 页)

(2)组合 2: $[(S5, S1), (S2, S7)]$. 先执行 $(S5, S1)$, 后执行 $(S2, S7)$. $S5$ 与 $S1$ 可并行. $S2$ 与 $S7$ 可并行.

(3)组合 3: $[(S1, S10), (S2, S6)]$. 先执行 $(S1, S10)$, 后执行 $(S2, S6)$. $S1$ 与 $S10$ 可并行. $S2$ 与 $S6$ 可并行.

5 结束语

文中提出了基于与或关系的 Web 服务组合方法. 通过生成 WSCM 以表达尽可能多的服务组合结果, 通过组合间的转化来适应网络环境的变化. 下一步的研究方向是在服务请求者端引入评估和优化机制, 实现多种服务组合的转换.

参考文献:

- [1] Xiaoning Ma, Baotian Dong, Ming He. AND/OR tree search algorithm in Web service composition[J]. PACIA, 2008(2):23-27.
- [2] Zhumin Chen, Jun Ma, Ling Song, et al. An efficient approach to Web services discovery and composition when

large scale services are available[J]. APSCC, 2006(1):34-41.

- [3] 刘安丰,陈志刚,桂卫华,等. 基于生成树的 Web 服务组合模型[J]. 计算机工程, 2008, 34(10): 108-110.
- [4] 李瑞宁,周竹荣. 基于完备回溯树的语义 Web 服务自动组合[J]. 计算机应用, 2008, 28(6): 1437-1430.
- [5] Haiming Tang, Farong Zhongt, Chang Yang. A tree-based method of web service composition[J]. ICPA, 2008(2):23-27.
- [6] 曹利培,刘静,缪淮扣. 基于图的 Web 服务组合优化的研究[J]. 计算机科学, 2007, 34(2): 95-99.
- [7] 龚小勇,朱庆生,武春岭,等. 基于参数推导图自动构建 Web 服务组合流程[J]. 计算机集成制造系统, 2008, 14(9): 1850-1855.
- [8] 陈瑜,苗克坚,李路明. 一种基于模板的服务组合方案[J]. 科学技术与工程, 2007, 7(10): 2273-2276.

作者简介:

李琦 女, (1980-), 硕士研究生. 研究方向为 Web 服务、网络计算、电子商务.

夏阳 男, (1962-), 博士, 教授. 研究方向为网络计算、电子商务.

(上接第 221 页)

参考文献:

- [1] Babcock B, Babu S, Datar Metal, Models and issue in data stream systems [C]// Proc. of 21th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems. Madison, 2002:1-16.
- [2] Mouratidis K, Bakiras S, Papadias D. Continuous monitoring of top-k queries over sliding windows[C]// Proc. Of ACM SIGMOD. USA:Chicago, 2006: 635-646.
- [3] Börzsönyi S, Kossmann D, Stocker K. The skyline operator[C]// ICDE (2001). Germany:Heidelberg, 2001.
- [4] Tao Y, Papadias D. Maintaining sliding window skylines on data streams[J]. IEEE Trans on KDE, 2006, 18(2):

377-391.

- [5] Man Lung Yiu. Nikos mamoulis: multi-dimensional top-k dominating queries[J]. The VLDB Journal, 2009(18): 695-718.

作者简介:

陈加付 男, (1985-), 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.