数据流上自适应的稀疏 Skyline 挖掘

苏 亮 [1] 邹 鹏 [1] 贾 焰 [1]

摘 要 Skyline 查询的结果集为数据集中不被其他对象所 "支配" 的对象的全体. 近年来, 它在在线服务、决策支持和实时监测等领域的良好应用前景, 使其成为数据管理与数据挖掘领域的研究热点. 实际应用中, 用户通常期望快速、渐进地获得 Skyline 计算结果, 而流数据的连续、海量、高维等特性, 使得在确保查询质量损失受控的前提下挖掘稀疏 Skyline 集合成为一个极具价值和挑战性的问题. 本文首先提出一个新颖的概念: 稀疏 Skyline (Sparse-skyline), 它采用一个 Skyline 对象来代表其周围 ε- 邻域内的所有 Skyline 对象; 接着, 给出了通过数据维度之间的相关性来自适应调整查询质量的两个在线算法; 最后, 理论分析和实验结果表明, 与现有的 Skyline 挖掘算法相比, 本文提出的方法具有良好的性能和效率, 更适合于数据流应用.

关键词 稀疏 Skyline, 自适应算法, 数据流, 数据挖掘
中图分类号 TP31

Adaptive Mining of Sparse Skyline over Data Stream

SU Liang[1] ZOU Peng[1] JIA Yan[1]

Abstract Skyline query set includes the objects that are not "dominated" by other objects in the dataset. In recent years, skyline query has been becoming a hot research topic due to its potential applications in online services, decision-making and real-time monitoring fields. Usually, people care about obtaining the skyline set quickly and progressively in real applications, however, because of the continuity, large-volume, and high-dimension of stream data, mining the sparse skyline set over data stream under control of losing quality is a more meaningful and challenging problem. In this paper, firstly, we propose a novel concept, called sparse-skyline, which uses a skyline object that represents its nearby skyline neighbors within ε-distance (acceptable difference). Then, two algorithms are developed which adopt correlation coefficient to adjust adaptively the quality of the sparse skyline query. Furthermore, theoretical analysis and experimental results show that the proposed methods are more efficient and effective compared with the existing skyline computing algorithm, and are suitable for data stream applications.

Key words Sparse skyline, adaptive algorithm, data stream, data mining

Skyline 查询也称作 Pareto 或极大向量问题 (Maximum vector problem[1−2]), Börzsönyi 等于 2001 年将其引入数据库查询处理领域[3]. 它是在给定对象集 Ω 中找出不被其他对象所 "支配" 的对象. 支配的含义为: d 维空间中的两个对象 $\boldsymbol{X} = (x_1, x_2, \cdots, x_d)$ 和 $\boldsymbol{Y} = (y_1, y_2, \cdots, y_d)$, 若对象 $\boldsymbol{X}$ 的任意维属性的评价值都不低于对象 $\boldsymbol{Y}$, 并且 $\boldsymbol{X}$ 至少存在一个属性的评价值高于 $\boldsymbol{Y}$, 则称 $\boldsymbol{X}$ 支配 $\boldsymbol{Y}$. 对象集 Ω 中不被任何其他对象所 "支配" 的对象就称为 Skyline 对象. Skyline 对象有一个非常重要的特性: 对于任意的单调递增函数 f: $\Omega \rightarrow \mathbf{R}$, 若 $\boldsymbol{X} \in \Omega$ 且 $\forall \boldsymbol{Y} \in \Omega, \boldsymbol{Y} \neq \boldsymbol{X}$, 有 $f(\boldsymbol{Y}) \geq f(\boldsymbol{X})$, 则 $\boldsymbol{Y}$ 属于 Skyline 集合[3]. 从集合论的角度看, Skyline 查询关注的是有限偏序集上极大元的集合. 由于 Skyline 集在许多应用中有着极其重要的意义, 它自从在文献 [1] 中被提出以来就一直是一个非常活跃的研究领域, 并且被广泛地应用于多约束决策支持[3−6], 数据挖掘和在线监控[7−10] 以及实时在线服务[11−14] 等方面. 例如, 在股票交易中, 股民通常关注两个因素: 风险和收益率. 股民在决定交易之前, 最想知道风险最小且收益率最高的股票. 然而, 在文献 [15] 中, Bentley 等已证明 Skyline 集合的平均数量为 $\mathrm{O}((\ln N)^{d-1})$ (N 为数据集 Ω 的元素数量, d 为每个元素的维数), 该理论得到了普遍的认可. 表 1 (见下页) 更能直观地展示: Skyline 查询的结果数目将随数据集大小 (N) 和维数 (d) 呈指数增长 (该数据由 Börzsönyi[5] 提供的源程序生成). 股民在面对众多支股票, 加上多方面决策因素 (如风险、收益率、投资期限等) 的情况下, 即便有一种算法和相应的硬件能实时并精确地计算出相应的 Skyline 集合, 也几乎不可能作出合理的决策, 其主要原因在于计算出来的 Skyline 集合数量实在太多. 不失一般性, Skyline 计算的目标条件可以由 min[3] 和 max[3] 算子组成, 但这两种算子在解决问题的方法上可以相互转化, 因此, 为简单起见, 本文假定 Skyline 计算的条件仅采用 min 算子组成.

收稿日期 2007-07-02 收修改稿日期 2007-09-29
Received July 2, 2007; in revised form September 29, 2007
国家高科技研究发展计划 (863 计划) (2006AA01Z451, 2006AA10Z237) 资助
Supported by National High Technology Research and Development Program of China (863 Program) (2006AA01Z451, 2006AA10Z237)
1. 国防科学技术大学计算机学院 长沙 410073
1. School of Computer Science, National University of Defense Technology, Changsha 410073
DOI: 10.3724/SP.J.1004.2008.00360

表 1　不同维数和数据分布下 Skyline 点集的大小
Table 1　Skylines in different dimensions and distributions

维数	相关分布	独立分布	反相关分布
2	1	13	51
3	3	74	641
4	12	286	4252
5	18	1106	12624
6	23	1997	26840
7	40	5560	41514
8	128	9664	55705
9	239	16857	67103
10	383	26062	75109

近年来, 一种新的应用模型 — 流数据 (Streaming data)[16] 模型广泛应用于众多领域, 例如: 金融应用[3]、网络监控[7]、兴趣趋势定位[16]、通信数据管理[5]、Web 应用、传感器网络数据处理[11]、电信欺诈探测[3] 等. 流数据一般具有如下特征: 数量大且速度快, 无限连续到达, 数据维度高, 并随时间具有不可预测性和多变性. 这些都给数据流挖掘领域的研究工作带来了许多新挑战. 针对数据流的这些特征, Kossmann 等在文献 [11] 中提出了数据流算法的一些准则, 可概括为如下 5 点:

1) 准确性: 结果符合查询要求;

2) 实时性: 能尽可能快地返回查询结果;

3) 普遍性: 算法不对数据的分布作预先假设;

4) 渐进性: 算法执行时间越长, 返回结果越多;

5) 可控性: 用户能对返回的结果根据自己的偏好进行有效地控制.

此前的 Skyline 研究工作大多作如下假设: 数据集更新不频繁或很少发生变化. 然而在现实中存在着大量不满足上述假设的情形, 如在线拍卖、股票推荐等基于动态数据的应用. 因此, 它们在现实应用中存在很大的局限性. 另一方面, 现有 Skyline 算法是获取精确的 Skyline 对象集合, 通过上述股票交易案例可知, 该精确 Skyline 集合数据量太大, 缺乏可控性, 很难快速捕获流数据的特征[11], 难以辅助决策者制定相应的策略. 如果存在另一种查询, 它返回大约一半甚至更少数量的 Skyline 对象, 而查询结果的质量却不受损失或者损失很小, 那么这种技术对决策者将具有更大的意义.

本文针对现有 Skyline 算法缺乏可控性以及数据流应用的需求 (特别是实时性和可控性), 展开了深入的研究. 据作者所知, 至今没有在数据流上进行稀疏 Skyline 计算的研究. 本文主要创新点如下:

1) 提出了一个新颖的概念: 稀疏 Skyline, 它用一个 Skyline 对象代表其周围 ε- 邻域 (可接受偏差) 内的所有 Skyline 对象;

2) 给出了采用可接受偏差进行剪枝和相关系数自适应调整稀疏查询质量的两个算法;

3) 通过理论分析和大量的实验, 验证了这两个算法的正确性和高可扩展性.

本文内容安排如下: 第 1 节介绍 Skyline 计算和数据流的相关工作; 第 2 节提出自适应稀疏 Skyline 问题和两个相应的算法; 第 3 节详细阐述实验的结果和性能评估; 第 4 节为总结和研究展望.

1　相关工作

自从 Börzsönyi 等[3] 在数据库领域提出 Skyline 计算问题以来, 它就一直是一个非常活跃的研究领域. 文献 [3] 给出了计算 Skyline 集合的两个算法: BNL (Block-nested loop) 和 DC (Divide and conquer) 算法. BNL 算法通过迭代, 将从数据集中读出的数据与内存中保存的候选 Skyline 对象作比较, 从而计算出所有的 Skyline 对象集; DC 算法先分割数据集, 然后合并每一部分计算出的局部的 Skyline 对象集, 从而得到整个数据集的 Skyline 集合. 自文献 [3] 之后, 大量的研究人员相继提出一系列的 Skyline 算法及其各种延伸的算法[4−14, 17−19], 它们主要针对算法的效率和不同的应用领域. 在文献 [11] 中, Kossmann 等结合最近邻搜索和 R-tree 索引数据的优点, 提出了最近邻 Skyline 计算方法, 提高了低维空间 Skyline 计算的性能. Chomicki 等[4] 通过对原有数据集进行预排序来提高 Skyline 查询的效率. 而文献 [5] 中的 BBS 算法则采用新发现的最近邻居对象对搜索空间进行剪枝, 并证明该算法的 I/O 性能最优. 更多的研究人员关注 Skyline 的特定应用领域和受限环境, 文献 [6] 延伸此问题到高维数据空间; Gao 等[13] 将该问题拓展到并行计算环境中; 文献 [7] 则考虑在分布环境下的快速 Skyline 计算; 文献 [10] 则关注 Peer-to-Peer 环境下的 Skyline 计算.

这些算法中, 文献 [4−6, 8−9, 11−12, 14, 17−18] 主要应用索引技术, 根据数据集中的各对象在所有维中属性值的关系, 剪掉一些明显不是 Skyline 的对象, 从而大大提高 Skyline 计算的效率. 基于主存的算法参见文献 [8−9, 11−12, 14, 18], 与数据库相关的技术参见文献 [4−14, 18−19]. 近年来, Skyline 研究转移到 Skyline 子空间 (Skyline cube, 或称 Subspace skyline)[8, 14, 18−19] 及数据流上的 Skyline[12] 计算方面.

上述算法都不支持数据流上的稀疏 Skyline 计算. 与本文最为接近的是文献 [12], Lin 等考虑对数据流中最近的 N 个元素计算其中的 Skyline 点集, 特别是它的 n-of-N Skyline 查询. 由于 Skyline 计算固有的特性, 该文作者意识到算法缺乏可控性, 难于适应决策支持等应用, 同时, 提出了在数据流上进行稀疏 Skyline 挖掘的问题. 本文在他们的研究

基础上, 对该问题进行了深入的研究, 给出了稀疏 Skyline 的形式化定义及两个有效的算法.

2 数据流上稀疏 Skyline 问题及算法

从表 1 可知, Skyline 查询的数量随对象集数目和维数呈指数增长. 此外, 数据流应用一般运行在存储容量有限 (相对于数据流的大小) 的环境中. 因此, 急需减少 Skyline 集合的数量, 改善算法的可控性. 从图 1 和图 2 中可见: 我们可以直接处理 Skyline 对象集, 适当地选取带圆圈的 Skyline 点, 去除这些点周围的 Skyline 点, 这些被去掉的 Skyline 点与一个或多个带圆圈的点很接近, 选取出来的带圆圈点集足以代表所有 Skyline 集合的整体特征. 因此, "相似性" 就成为衡量稀疏程度的重要指标.

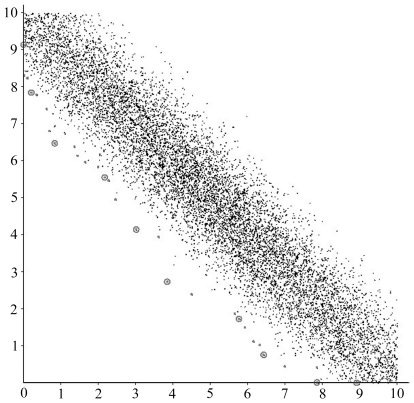

图 1 反相关分布

Fig. 1 Anti-correlation distribution

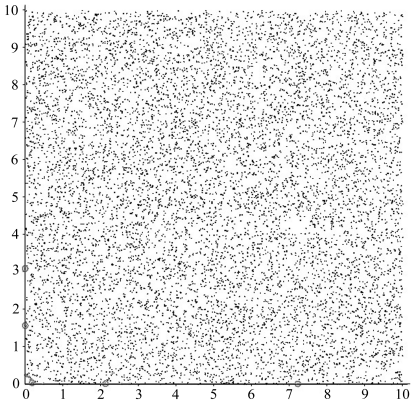

图 2 独立分布

Fig. 2 Independent distribution

为了精确地描述两个点相似的程度, 很自然地会选择欧氏距离 (d- 维度量空间 V), 该空间关联一个距离度量函数 dis. 该函数必须具备如下性质:

1) $dis(\boldsymbol{X}_i, \boldsymbol{X}_j) = dis(\boldsymbol{X}_j, \boldsymbol{X}_i)$;

2) $dis(\boldsymbol{X}_i, \boldsymbol{X}_j) \geq 0$, 且若 $\boldsymbol{X}_i = \boldsymbol{X}_j$, 则 $dis(\boldsymbol{X}_i, \boldsymbol{X}_j) = 0$;

3) $dis(\boldsymbol{X}_i, \boldsymbol{X}_k) \leq dis(\boldsymbol{X}_i, \boldsymbol{X}_j) + dis(\boldsymbol{X}_j, \boldsymbol{X}_k)$.

普遍使用的距离函数有欧几里得距离和曼哈顿距离. 不失一般性, 本文仅使用欧几里得距离作为 dis 函数, 即

$$dis(\boldsymbol{X}_i, \boldsymbol{X}_j) = \sqrt{\sum_{k=1}^{d} (x_i^k - x_j^k)^2} \tag{1}$$

定义 1. 稀疏 Skyline 集合. 令 Ω 为 d 维数据集, Ω_S 为 Ω 上的 Skyline 集合, 可接受的偏差距离为 ε, 若集合 Ω_A 满足如下条件:

1) $\Omega_A \subseteq \Omega_S$;

2) $\forall \boldsymbol{X}_i, \boldsymbol{X}_j \in \Omega_A$, $\boldsymbol{X}_i \neq \boldsymbol{X}_j$, 则 $dis(\boldsymbol{X}_i, \boldsymbol{X}_j) > \varepsilon$;

3) $\forall \boldsymbol{X}_i \in \Omega_S$, 则 $\exists \boldsymbol{X}_j \in \Omega_A$, 使 $dis(\boldsymbol{X}_i, \boldsymbol{X}_j) \leq \varepsilon$.

则称 Ω_A 为稀疏 Skyline 集合.

定义 1 中的条件 1) 表明稀疏 Skyline 集合为 Skyline 集合的子集; 条件 2) 主要是保证稀疏 Skyline 集合中对象的稀疏程度, 确保不存在任意两个对象的距离大于可接受的偏差距离; 满足条件 3) 则体现稀疏 Skyline 集合能在可接受的偏差距离内展示 Skyline 集合的整体特征. 上述三个条件缺一不可.

定义 2. 强支配关系. 对象 $\boldsymbol{X}_i, \boldsymbol{X}_j \in \Omega$, $\boldsymbol{X}_i$ 支配 $\boldsymbol{X}_j$ 记作 $\boldsymbol{X}_i \rhd \boldsymbol{X}_j$, $\boldsymbol{X}_i$ 强支配 $\boldsymbol{X}_j$ 记作 $\boldsymbol{X}_i + \varepsilon \succ \boldsymbol{X}_j$. 即: $\forall 1 \leq k \leq d$, $x_i^k + \varepsilon \leq x_j^k$, 且 $\exists 1 \leq t \leq d$, $x_i^t + \varepsilon < x_j^t$.

定义 3. 相关系数. 对于 n 个 d 维对象 $\boldsymbol{X}_i = (x_i^1, x_i^2, \cdots, x_i^d)$, $1 \leq i \leq n$, 设矩阵 $A = (\boldsymbol{X}_1, \boldsymbol{X}_2, \cdots, \boldsymbol{X}_n)^{\mathrm{T}}$ 和向量 $\boldsymbol{Y}_j = (x_1^j, x_2^j, \cdots, x_n^j)^{\mathrm{T}}$, 则 $A = \begin{pmatrix} x_1^1 & \cdots & x_1^d \\ \vdots & \ddots & \vdots \\ x_n^1 & \cdots & x_n^d \end{pmatrix} = (\boldsymbol{Y}_1, \cdots, \boldsymbol{Y}_d)$, 且 $\rho(\boldsymbol{Y}_i, \boldsymbol{Y}_j) = \dfrac{\mathrm{E}((\boldsymbol{Y}_i - \mathrm{E}\boldsymbol{Y}_i)(\boldsymbol{Y}_j - \mathrm{E}\boldsymbol{Y}_j))}{\sqrt{\mathrm{D}\boldsymbol{Y}_i} \cdot \sqrt{\mathrm{D}\boldsymbol{Y}_j}} = \dfrac{\mathrm{E}(\boldsymbol{Y}_i \boldsymbol{Y}_j) - \mathrm{E}\boldsymbol{Y}_i \mathrm{E}\boldsymbol{Y}_j}{\sqrt{\mathrm{D}\boldsymbol{Y}_i} \cdot \sqrt{\mathrm{D}\boldsymbol{Y}_j}}$ 为 $\boldsymbol{Y}_i$ 和 $\boldsymbol{Y}_j$ 的相关系数, 其中, $\mathrm{E}\boldsymbol{Y}_i = (\sum_{k=1}^{n} x_k^i)/n$ 是向量 $\boldsymbol{Y}_i$ 的期望; $\mathrm{D}\boldsymbol{Y}_i = \mathrm{E}(\boldsymbol{Y}_i - \mathrm{E}\boldsymbol{Y}_i)^2 = \mathrm{E}\boldsymbol{Y}_i^2 - (\mathrm{E}\boldsymbol{Y}_i)^2$ 是向量 $\boldsymbol{Y}_i$ 的方差.

相关系数的值在 -1 和 $+1$ 之间. 当 $\rho = -1$ 时, $\boldsymbol{Y}_i$ 与 $\boldsymbol{Y}_j$ 负线性相关; $\rho = 0$ 时, $\boldsymbol{Y}_i$ 与 $\boldsymbol{Y}_j$ 不相关; $\rho = 1$ 时, $\boldsymbol{Y}_i$ 与 $\boldsymbol{Y}_j$ 正线性相关. 这三个值与反相关、独立和相关数据分布相对应[3]. 通过对表 1 中数据的分析得知, 数据分布对 Skyline 结果有巨大影响, 特别是反相关与独立分布. 由于 $\rho \in [-1, 1]$, 为了便于算法的统一处理, 所以我们构造转换函数

$$\Phi(\rho) = (a + b)/2 - \rho(b - a)/2, \quad 1 \leq a \leq b$$

一般 a 取 1, b 取 2 或大于 2 的其他值. 如果 b 大于 2, 则相关系数的影响大于 ε, 因此我们选择 $\varepsilon\Phi(\rho)$

作为偏差距离的限制, 在数据流上自适应地调整稀疏 Skyline 计算的结果. 下面定理 1 保证了在可接受差异距离内 Skyline 集合的界限, 应用定理 1 的逆否命题可对稀疏 Skyline 集合的计算进行快速剪枝.

定理 1. 如果 $\boldsymbol{X}_i, \boldsymbol{X}_j \in \Omega_A$ 且 $dis(\boldsymbol{X}_i, \boldsymbol{X}_j) \le \varepsilon$, 则 $\sum_{k=1}^{d} \left|x_i^k - x_j^k\right| \le \varepsilon\sqrt{d}$.

证明. 由式 (1) 得知: $(dis(\boldsymbol{X}_i, \boldsymbol{X}_j))^2 = \sum_{k=1}^{d}(x_i^k - x_j^k)^2 \le \varepsilon^2$, 令 $\left|x_i^k - x_j^k\right| = P_k$, 那么,

$$\begin{aligned}(d-1)\sum_{k=1}^{d}(P_k)^2 &= \sum_{1\le i<j\le d}\left[(P_i)^2 + (P_j)^2\right] \\ &\ge 2\sum_{1\le i<j\le d}(P_i \cdot P_j)\end{aligned}$$

因此,

$$\sum_{k=1}^{d}\left|x_i^k - x_j^k\right| = \sum_{k=1}^{d} P_k = \sqrt{(\sum_{k=1}^{d} P_k)^2} = \sqrt{\sum_{k=1}^{d}(P_k)^2 + 2\sum_{1\le i<j\le d}(P_i \cdot P_j)} \le \sqrt{\varepsilon^2 + (d-1)\varepsilon^2} = \varepsilon\sqrt{d} \qquad \square$$

2.1 Naïve 稀疏 Skyline 算法

通常, 系统中只能获得原始数据集 Ω. 由定义 1 可知, 稀疏 Skyline 集合是 Skyline 集合的子集, 一种 Naïve 方法为: 首先, 计算精确的 Skyline 集合, 然后过滤那些相似的对象, 进而获得稀疏 Skyline 集合. 在第一阶段, 采用文献 [12] 提供的基于滑动窗口的 Stabling Skyline 算法 (SSA 算法), 将其结果保存在 SkylineList 数据结构中. 接着, 对 SkylineList 进行排序, 在定理 1 的保证下尽可能早地停止迭代. NAS 算法 (Naïve approximate skyline algorithm) 的伪代码参见 Algorithm 1.

Algorithm 1 $\boldsymbol{NAS}(\varepsilon, \Omega_S)$

Input: ε 为稀疏 Skyline 计算的差异距离; **SkylineList** 保存精确的 Skyline 对象; **SortList** 函数使用单调函数 $(\sum_{i=1}^{d} x_i)$ 对所有 Skyline 对象按降序排列.

Output: 算法输出为稀疏 Skyline 集 Ω_A.

```
1  SkylineList = SortList(Ω_S);
2  var current = 0; var second = 0;
3  for (current = 0; current < SkylineList.count; current++) do
4     for (second = current + 1; second < SkylineList.count; second++) do
5        if (dis(SkylineList[second], SkylineList[current]) ≤ εΦ(ρ)) then
6           SkylineList.delete(second);
7        end if
8        if (dis(SkylineList[second], SkylineList[current]) > ε√d) then
9           break;
10       end if
11    end for
12 end for
```

2.2 基于扩展 Micro-cluster 特性树的稀疏 Skyline 算法

虽然 NAS 算法可行且有效, 但是存在两个主要问题: 1) 它必须在内存中保存精确的 Skyline 集合; 2) 当一个新数据到达时, 需要重新执行 NAS 算法以获得稀疏 Skyline 集合. 由于在数据流应用中内存受限, 且需近实时地得到相应的结果, 因此, NAS 算法的这两个缺陷使得它不适合数据流模型. 为了解决这些问题, 一种常用的策略是进行数据压缩或摘要, 这样既能节省内存, 也能提高算法的运行效率. 因此我们采用 CF-tree[20] 对数据流滑动窗口内数据进行划分, 表示为扩展 Micro-cluster 特性树, 其中 CF-tree 只保存稀疏 Skyline 集合以及与之邻近节点的信息 (两个或更多的 Micro-cluster 可以构建一个更大的 Micro-cluster, 直至最后滑动窗口中的所有对象构成一棵 Micro-cluster tree).

定义 4. 扩展 Micro-cluster 特性树 (EMCF-tree). d 维对象 $\boldsymbol{X}_1, \cdots, \boldsymbol{X}_n$ ($\boldsymbol{X}_i = (x_i^1, \cdots, x_i^d)$, $i = 1, \cdots, n$) 的一个扩展 Micro-cluster 特性树定义为: 一个长度为 $(3d + 3 + L(H_{Skyline}) + L(H_{dominated}))$ 的元组, 元组中每个元素是 ($\overline{\boldsymbol{ASF1}}$, $\overline{\boldsymbol{ASF2}}$, $\overline{\boldsymbol{ASF3}}$, $K(\boldsymbol{X}_{centriod})$, $\boldsymbol{X}_{centriod}$, $Parent$, $H_{Skyline}$, $H_{dominated}$). 每个输入的定义如下 (图 3 为 EMCF-tree 的整个数据结构):

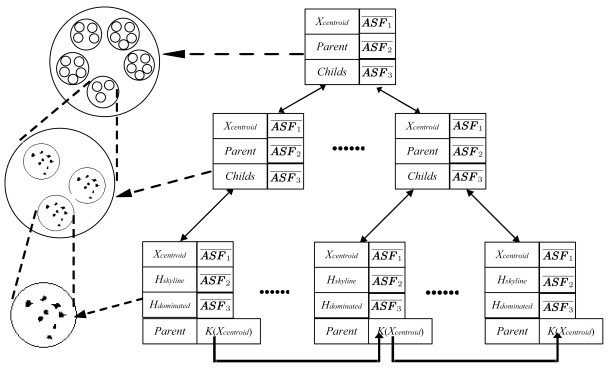

图 3　扩展的 Micro-cluster 特性树

Fig. 3　Extended Micro-cluster feather tree

1) 向量 $\overline{\boldsymbol{ASF1}}$ 的第 p 项为 $\sum_{j=1}^{n} x_j^p$; 向量 $\overline{\boldsymbol{ASF2}}$ 的第 p 项为 $\sum_{j=1}^{n}(x_j^p)^2$; 向量 $\overline{\boldsymbol{ASF3}}$ 的第 p 项为 $\min_{j=1}^{n}(x_j^p)$.

2) $\boldsymbol{X}_{centriod}$ 属于 Ω_A, 代表一个 Micro-cluster 的质心; $K(\boldsymbol{X}_{centriod})$ 为 $\boldsymbol{X}_{centriod}$ 的秩; $Parent$ 为指向其父结点的指针, 用于向上回溯修改该节点的

祖先结点.

3) $H_{Skyline}$ 是一个堆 (Heap), 保存 Micro-cluster 中 Skyline 对象; $H_{dominated}$ 是一个堆, 保存被 $\boldsymbol{X}_{centriod}$ 支配的对象; n 是滑动窗口的长度.

该扩展的 Micro-cluster 数据结构继承了原有数据结构的易于更新的特点. 整个算法描述参见 Algorithm 2.

Algorithm 2 ***EMCFTA***(ε, Ω)

Input: ε 为稀疏 Skyline 计算的差异距离; 输入为 Ω 中的原始数据对象; $\boldsymbol{X}_{new}$ 表示一个数据流中新到的数据对象.

Output: 算法输出为稀疏 Skyline 集 Ω_A.

从 EMCF-tree 的根节点出发, 查找离 $\boldsymbol{X}_{new}$ 最近的 Micro-cluster (称为 MC_d), 它必为该树的一个叶节点;
if $(\boldsymbol{X}_{new} \succ \boldsymbol{X}_{centriod})$ **then**
  $\boldsymbol{X}_{new}$ 处于图 4 的区域 (V), 因此 $\boldsymbol{X}_{centriod} = \boldsymbol{X}_{new}$;
else if $(\boldsymbol{X}_{centriod} \succ \boldsymbol{X}_{new})$ **then**
  begin
  if $(K(\boldsymbol{X}_{new}) - K(\boldsymbol{X}_{centriod}) \leq n - K(\boldsymbol{X}_{centriod}) \mod n)$ **then**
    不作任何处理;
  else
    $\boldsymbol{X}_{new}$ 处于图 4 的区域 (I), 因此将 $\boldsymbol{X}_{new}$ 添加到 $H_{dominated}$ 中;
  end if
  end;
  /*$\boldsymbol{X}_{new}$ 和 $\boldsymbol{X}_{centriod}$ 都是 Skyline 对象 */
else if $(dis(\boldsymbol{X}_{new}, \boldsymbol{X}_{centriod}) \leq \varepsilon)$ **then**
  $\boldsymbol{X}_{new}$ 处于图 4 的区域 (IV), 将 $\boldsymbol{X}_{new}$ 添加到 $H_{Skyline}$;
else
  $\boldsymbol{X}_{new}$ 处于图 4 的区域 (III), 创建一个新的 Micro-cluster 叶节点, 其父节点为 MC_d, 并将 $\boldsymbol{X}_{new}$ 作为该叶节点的中心.
end if

EMCFTA 算法 (Extended micro-cluster feature tree algorithm) 的主要思想如下: 1) 随着数据到达, 算法构造扩展 Micro-cluster 特性树, 同时, 从根结点开始寻找最近的 Micro-cluster, 它是伴随新进入对象的叶结点; 2) 从图 4 比较 $\boldsymbol{X}_{centriod}$ 和 $\boldsymbol{X}_{new}$ 时, 可以清楚地观察到新进入对象只有 5 种可能的结果, EMCFTA 算法分别对它们进行处理.

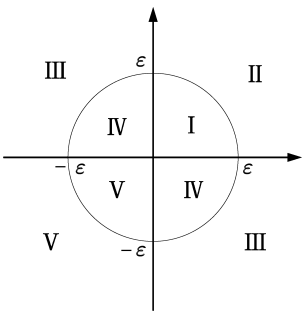

图 4 Micro-cluster 区域

Fig. 4 Micro-cluster regions

3 实验结果与分析

据作者所知, 至今没有在数据流上进行稀疏 Skyline 计算的研究. 实验上, 我们用 Visual Studio C++ 2003 实现了一个仿真系统, 比较 NAS 算法、EMCFTA 算法与 Lin 等[12] 提出的 SSA (Stabling Skyline algorithm) 算法的准确性和效率. 实验运行环境为 Intel Pentium 4, 2 GHz CPU, 512 MB RAM, Windows XP. 实验中的数据集采用 Skyline 计算领域通行的模拟数据集: 相关、独立和反相关数据集[4−14, 17−19]. 通常真实数据可以被看作是上述三种典型数据分布的组合. Börzsönyi[3] 提供了产生这三种典型数据分布的源程序, 生成这些数据分布的细节参见文献 [3]. 所有生成的数据值介于 0 和 1 之间, 偏差距离 $\varepsilon \in [0, 1]$, 维数 d 取值 2 到 10 之间.

3.1 算法的可管理性

该实验比较在三种数据分布情况下的 Ω_A 与 Ω_S 集合的大小. Ω_A 集合由 EMCFTA 算法生成, Ω_S 集合由 SSA 算法产生. 实验表明 Ω_A 集合的大小将随着 Ω 集合的大小和维数的增加而增加 (见图 5). 在 $d = 5$ 和 $\varepsilon = 0.1$ 的情况下, 由 EMCFTA 算法生成的 Ω_A 结果集大小为 Ω_S 集合的 30 % 到 55 % 之间. 如果增大偏差距离, Ω_A 结果集将小于 Ω_S 集合的 40 %. 特别是, 反相关数据分布减少得更多, 由于自适应参数 $\varepsilon\Phi(\rho)$ 的存在, 实验证明稀疏 Skyline 算法能迅速检测到数据的分布, 并自适应地调整稀疏 Skyline 集合中的对象数, 以充分利用系统的资源 (如: 计算资源和存储资源等).

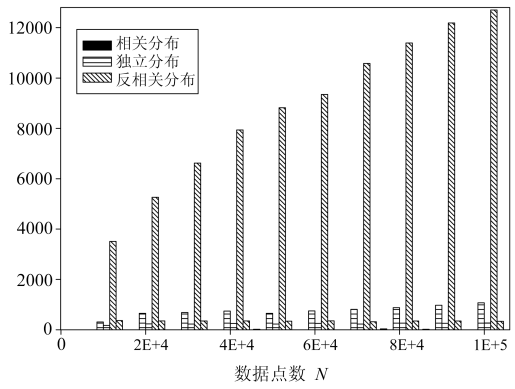

图 5 Ω_A 与 Ω_S 的比较: $d = 5$, $\varepsilon = 0.1$

Fig. 5 Comparison of Ω_A and Ω_S: $d = 5$, $\varepsilon = 0.1$

3.2 稀疏Skyline集合的质量

本实验中使用的数据集和参数与前一实验相同. Ω_A 集合由 EMCFTA 算法生成, Ω_S 集合由 SSA 算法产生. 由图 5 可知, 相关分布中的 Ω_S 集合的数目明显少于其他两种分布, 因此仅考虑反相关分布和独立分布. 为了度量 Ω_A 集合的质量, 我们

分别定义了 Ω_A 和 Ω_S 中任意两个对象的平均距离: $AvgDis_A$ 和 $AvgDis_S$ (参见式 (2) 和 (3), 其中 $|\Omega_S|$、$|\Omega_A|$ 分别为集合 Ω_S 和 Ω_A 的元素数量), 通过平均距离的差异来衡量稀疏 Skyline 计算与精确 Skyline 计算获得的结果之间的质量差异. 图 6 中, 先固定维数为 5, ε 为 0.15, 在此条件下, 检测数据流中滑动窗口的大小 (n) 对计算质量的影响. 由图 6 可知, 平均距离比率 ($100\times AvgDis_S/AvgDis_A$) 处于 87 和 99 之间, 而稀疏 Skyline 对象数量已经减少了 45 %. 这就是说, 稀疏 Skyline 计算能在只降低大约 10 % 的质量下, 却减少了几乎一半的 Skyline 对象, 极大地提高了数据的可管理性. 由于滑动窗口都是处于主存中, 并且 EMCFTA 算法和 SSA 算法都是基于主存的算法, 因此, 滑动窗口的大小对稀疏 Skyline 结果集的质量没有太大的影响. 接下来, 固定滑动窗口的大小为 100 000, 考察参数 ε 对质量的影响, 从图 7 可得, 平均距离比率在 61 和 96 之间, 在 16 次测试中, 有 12 次的平均距离比率超过 85, 小于 75 的只有 2 次测试, 16 次测试的平均比率为 89.6. 通过这三类测试, 充分表明 EMCFTA 算法能在确保 Skyline 结果集质量的前提下, 极大地提高结果集的可管理性, 增强了决策的效率.

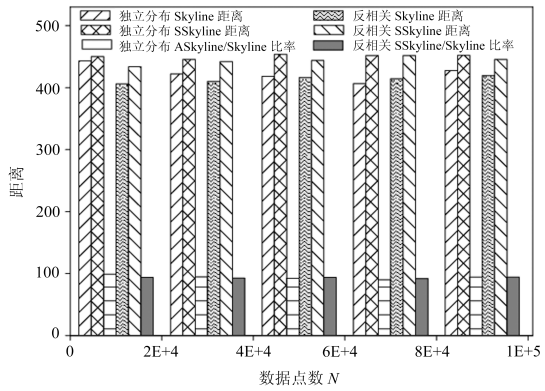

图 6　平均距离: $d=5$, $\varepsilon=0.15$

Fig. 6　Average distance: $d=5$, $\varepsilon=0.15$

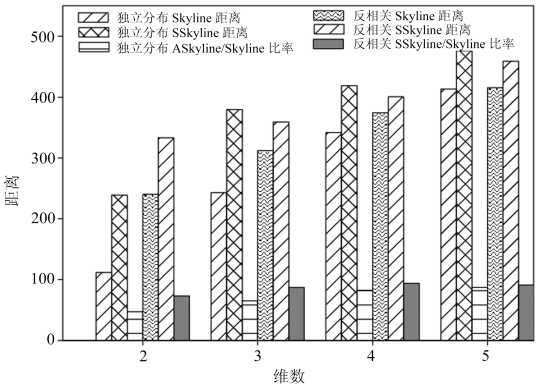

图 7　平均距离: $\varepsilon=0.2$, $n=100\,000$

Fig. 7　Average distance: $\varepsilon=0.2$, $n=100\,000$

$$AvgDis_S=\frac{\sum_{i=1}^{|\Omega_S|}(\sum_{j=2\wedge j\neq i}^{|\Omega_S|}dis(X_i,X_j))}{|\Omega_S|\cdot(|\Omega_S|-1)/2}\tag{2}$$

$$AvgDis_A=\frac{\sum_{i=1}^{|\Omega_A|}(\sum_{j=2\wedge j\neq i}^{|\Omega_A|}dis(X_i,X_j))}{|\Omega_A|\cdot(|\Omega_A|-1)/2}\tag{3}$$

3.3　算法运行效率

本实验比较 NAS 和 EMCFTA 两种算法的效率, 数据集和参数依然与前面的实验相同. 我们比较每种算法在不同参数设置情况下耗费的总时间. 图 8 显示了在 $\varepsilon=0.15$ 和 $d=2$ 时的运行时间, 由于只有两维, 使用 NAS 和 EMCFTA 算法计算点之间的距离都比较快, 从而 EMCFTA 与 NAS 相比在不同的滑动窗口大小下只有 1 %～8 % 的性能提升. 在图 9 中, 由于维数上升为 5, 可以观察到 EMCFTA 算法明显快于 NAS 算法, 快大约 10 %～18 %, 特别是在反相关数据分布下. 由此可知, EMCFTA 算法更适合于大数量和频繁更新的数据流环境.

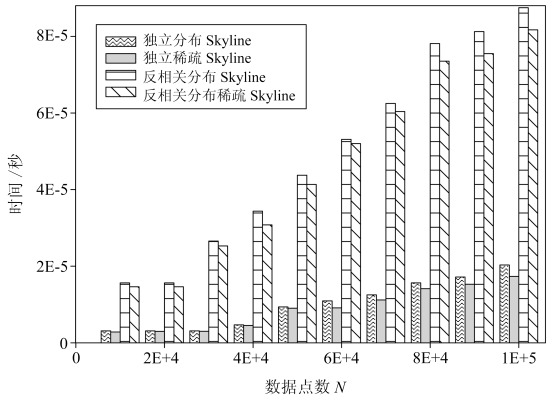

图 8　运行时间比较: $\varepsilon=0.15$, $d=2$

Fig. 8　Comparison of runtime: $\varepsilon=0.15$, $d=2$

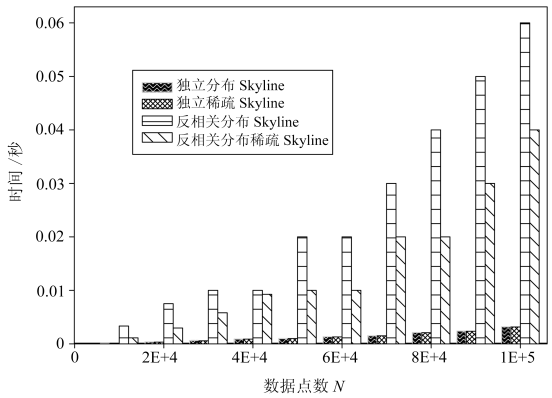

图 9　运行时间比较: $\varepsilon=0.15$, $d=5$

Fig. 9　Comparison of runtime: $\varepsilon=0.15$, $d=5$

4 总结与展望

本文研究了具有重要实际应用价值的问题 — 稀疏 Skyline 挖掘, 并最先提出了计算稀疏 Skyline 的两个新颖高效的算法, 它们利用数据维度之间的相关系数进行自适应地剪枝, 确保稀疏 Skyline 集合的质量和可控性. 理论分析和实验结果表明, 本方法在尽量不降低质量的前提下, 极大地减少了算法的存储空间和运行时间, 能应用于实时数据流分析. 下一步的研究将在现有自适应的稀疏 Skyline 挖掘算法的基础上, 将其拓展到分布式数据流环境, 支持更大并行和分布的稀疏 Skyline 挖掘.

References

1 Kung H T, Luccio F, Preparata F P. On finding the maxima of a set of vectors. *Journal of the Association for Computing Machinery*, 1975, **22**(4): 469−476

2 Preparata F P, Shamos M I. *Computational Geometry: An Introduction*. Berlin: Springer, 1985

3 Börzsönyi S, Kossmann D, Stocker K. The skyline operator. In: Proceedings of the 17th International Conference on Data Engineering. Heidelberg, Germany: IEEE, 2001. 421−430

4 Chomicki J, Godfrey P, Gryz J, Liang D. Skyline with presorting. In: Proceedings of the 19th International Conference on Data Engineering. Toronto, Canada: IEEE, 2003. 717−719

5 Papadias D, Tao Y F, Fu G, Seeger B. An optimal and progressive algorithm for skyline queries. In: Proceedings of ACM SIGMOD International Conference on Management of Data. San Diego, USA: ACM, 2003. 467−478

6 Chan C Y, Jagadish H V, Tan K L, Tung A K H, Zhang Z. On high dimensional skylines. In: Proceedings of International Conference on Extending Database Technology. Crete, Greece: Springer, 2006. 478−495

7 Deng Bo, Jia Yan, Yang Shu-Qiang. An efficient algorithm for distributed skyline queries. *Computer Engineering and Science*, 2007, **29**(9): 97−100
(邓波, 贾焰, 杨树强. 一种高效的分布式的 Skyline 查询算法. 计算机工程与科学, 2007, **29**(9): 97−100)

8 Brando C, Goncalves M, González V. Evaluating top-k skyline queries over relational databases. In: Proceedings of International Conference on Database and Expert Systems Applications. San Jose, USA: Springer, 2007. 254−263

9 Vlachou A, Doulkeridis C, Kotidis Y, Vazirgiannis M. SKYPEER: efficient subspace skyline computation over distributed data. In: Proceedings of the 23rd International Conference on Data Engineering. Istanbul, Turkey: IEEE, 2007. 416−425

10 Wang S Y, Ooi B C, Tung A K H, Xu L Z. Efficient skyline query processing on peer-to-peer networks. In: Proceedings of the 23rd International Conference on Data Engineering. Istanbul, Turkey: IEEE, 2007. 1126−1135

11 Kossmann D, Ramsak F, Rost S. Shooting stars in the sky: an online algorithm for skyline queries. In: Proceedings of the 28th International Conference on Very Large Databases. Hong Kong, China: ACM, 2002. 275−286

12 Lin X M, Yuan Y D, Wang W, Lu H Q. Stabbing the sky: efficient skyline computation over sliding windows. In: Proceedings of the 21st International Conference on Data Engineering. Sydney, Australia: IEEE, 2005. 502−513

13 Gao Y J, Chen G C, Chen L, Chen C. Parallelizing progressive computation for skyline queries in multi-disk environment. In: Proceedings of International Conference on Database and Expert Systems Applications. Springer, 2006. 697−706

14 Xia T, Zhang D. Refreshing the sky: the compressed skycube with efficient support for frequent updates. In: Proceedings of SIGMOD. Chicago, USA: ACM, 2006. 491−502

15 Bentley J L, Kung H T, Schkolnick M, Thompson C D. On the average number of maxima in a set of vectors and applications. *Journal of the Association for Computing Machinery*, 1978, **25**(4): 536−543

16 Babcock B, Babu S, Datar M, Motwani R, Widom J. Models and issues in data stream systems. In: Proceedings of the 21st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems. Madison, USA: ACM, 2002. 1−16

17 Luo Y, Lu H X, Lin X M. A scalable and I/O optimal skyline processing algorithm. In: Proceedings of International Conference on Web-age Information Management. Dalian, China: Springer, 2004. 218−228

18 Pei J, Fu A W C, Lin X M, Wang H X. Computing compressed multidimensional skyline cubes efficiently. In: Proceedings of the 23rd International Conference on Data Engineering. Istanbul, Turkey: IEEE, 2007. 96−105

19 Tao Y F, Xiao X K, Pei J. Subsky: efficient computation of skylines in subspaces. In: Proceedings of the 22nd International Conference on Data Engineering. Atlanta, USA: IEEE, 2006. 65

20 Zhang T, Ramakrishnan R, Livny M. Birch: an efficient data clustering method for very large databases. In: Proceedings of SIGMOD International Conference on Management of Data. Montreal, Canada: ACM, 1996. 103−114

苏 亮 国防科学技术大学博士研究生. 主要研究方向为分布计算和数据挖掘. 本文通信作者.
E-mail: suliangnudt@gmail.com
(**LIANG Su** Ph. D. candidate at National University of Defense Technology. His research interest covers distributed computing, database, and data mining. Corresponding author of this paper.)

邹 鹏 国防科学技术大学教授. 主要研究方向为分布计算, 操作系统.
(**PENG Zou** Professor, Ph. D. at National University of Defense Technology. His research interest covers distributed computing and operating system.)

贾 焰 国防科学技术大学教授. 主要研究方向为分布计算, 数据库.
(**JIA Yan** Professor, Ph. D. at National University of Defense Technology. Her research interest covers distributed computing and database.)