一种基于索引的高效 k-支配 Skyline 算法

印 鉴 姚树宇 薛少锷 杨文新 刘玉葆
(中山大学信息科学与技术学院 广州 510006)

摘 要 由于在多标准决策支持等应用上具有重要的意义,skyline 查询成为近年来数据库和数据挖掘领域的一个研究热点.然而随着数据集维数的增加,数据点之间形成支配关系的可能性越来越小,导致了 skyline 点数目过多而无法提供任何有效信息.为了在高维数据集中找到更重要和更有意义的 skyline 点,人们提出了 k-支配 skyline 的定义.但现有的用于 k-支配 skyline 的算法在时间效率、空间复杂度和渐进输出性上都有待提高.该文提出了一种基于索引的高效 k-支配 skyline 算法,通过为数据集建立两个索引,算法可以高效地进行计算,在时间、空间和渐进性上均优于现有的算法.

关键词 skyline;决策支持;k-支配 skyline;基于索引
中图法分类号 TP311 **DOI 号**: 10.3724/SP.J.1016.2010.01236

An Index Based Efficient k-Dominant Skyline Algorithm

YIN Jian YAO Shu-Yu XUE Shao-E YANG Wen-Xin LIU Yu-Bao
(*School of Information Science and Technology, Sun Yat-Sen University, Guangzhou* 510006)

Abstract Due to the importance for several applications involving multi-criteria decision making, skyline query has received a lot of attention in the research field of database and data mining in recent years. However, as the number of dimensions increase, the possibility to form dominant relationship between data points is very low. As a result, the number of skyline points becomes too numerous to provide any useful information. For the sake of finding more important and more meaningful skyline points in high dimension data set, a new concept called k-dominant skyline was proposed. Currently the algorithms used for k-dominant skyline query do not have good performance on the time and space complexity. None of them is a good online algorithm. This paper proposes a new index based algorithm, by building two index of the data set, the algorithm can efficiently compute k-dominant skyline. It is an online algorithm and has better performance than all of current algorithms.

Keywords skyline; decision support; k-dominant skyline; index based

1 引 言

Skyline 查询是从数据库中抽取有趣信息以帮助用户做出决策的一个新的应用,由 Borzsonyi、Kossmann、Stocker 等在 2001 年的 ICDE 上提出[1],近年来已成为数据库和数据挖掘领域的研究热点之一.Skyline 查询从数据集中抽取不被其他任

收稿日期:2008-10-08;最终修改稿收到日期:2010-05-21.本课题得到国家自然科学基金(60773198,60703111)、广东省自然科学基金(7300272,8151027501000021)、国家科技计划项目(2008ZX10005-013)、广东省科技计划项目(2008B050100040,2009A080207005,2009B090300450)、新世纪优秀人才支持计划(NCET-06-0727)资助.**印 鉴**,男,1968 年生,教授,博士生导师,主要研究领域为数据库、数据挖掘、人工智能等.E-mail: issjyin@mail.sysu.edu.cn.**姚树宇**,男,1983 年生,硕士研究生,主要研究方向为数据库与数据挖掘.**薛少锷**,男,1983 年生,硕士研究生,主要研究方向为数据库与数据挖掘.**杨文新**,男,1984 年生,硕士研究生,主要研究方向为数据库与数据挖掘.**刘玉葆**,男,1975 年生,副教授,主要研究方向为数据库与数据挖掘.

何数据点支配(dominate)的数据点集合. 一个数据点 p 支配另一个数据点 q，当且仅当点 p 在所有维上的取值都不比 q 差，并且在至少一个维上的值比 q 好. 而一个多维数据集的 skyline 则是该数据集上不被任何其它数据点支配的点所组成的集合.

在实际应用中，查询对象往往是高维数据集，随着数据集维数的增加，一个点支配另一个点的概率变得非常小，这导致计算出来的 skyline 数据集规模太大，甚至接近原数据集的大小，无法提供给用户有意义的信息. 为了在高维空间中寻找出更重要和更有意义的 skyline 点集，Chan 等[2]提出了在数据集中计算 k-支配 skyline 的概念. 通过弱化“支配”的定义，来提高数据点之间形成“k-支配”关系的概率，使得查询出来的结果数据集规模更小，具有更好的决策支持作用.

由于 k-支配 skyline 的特殊性，传统 skyline 算法[3-7]等不适用于 k-支配 skyline 查询. 目前在 k-支配 skyline 的计算上主要有 3 种算法，在时间效率、空间复杂度和渐进性上都存在不足. 本文提出一种基于索引的高效算法，通过预处理建立数据集的两个索引，在计算过程中依靠索引让数据集中最有可能属于 k-支配 skyline 的数据点被优先处理，迅速返回部分查询结果，而不属于结果的数据点则能尽快被淘汰掉，减少多余的比较，从而使算法在时间效率和渐进输出性上有较大提高，与此同时算法还降低了计算需要的空间开销. 实验结果表明其性能比现有 3 种算法都好.

本文第 2 节介绍 k-支配 skyline 的定义；第 3 节介绍现有 3 种算法的基本思想以及存在的不足；第 4 节提出基于索引的 k-支配 skyline 算法，介绍算法的基本思路和具体实现；第 5 节是对本文提出的算法与现有 3 种算法进行实验比较和分析；第 6 节是小结.

2　概念描述

在解决高维空间中 skyline 数据集过大的问题上，目前有几种主要的方法，如文献[8-9]通过挖掘更严格定义的 skyline 来降低结果数据集的规模. 更为直观和有意义的是 k-支配 skyline 的挖掘，本节将首先介绍 k-支配 skyline 的相关定义，下一节介绍并分析现有算法的缺点.

给定一个 d 维空间 $S=\{s_1,s_2,\cdots,s_d\}$，称数据集 $D=\{p_1,p_2,\cdots,p_d\}$ 是 S 上的数据集，如果 D 中的每一个数据点 $p_i\in D$ 是 S 空间中的一个 d 维数据点.

用 $p_i.s_j$ 表示数据点 p_i 的第 j 个维度的值. 对于每个维度 s_i，假设在它的整个值域上存在一个全序关系，表示为 $\prec_i$. 这里 $\prec_i$ 根据不同情况或者用户的不同选择可以是 $>$ 或 $<$ 的关系. 为了简便，不失一般性，本文后面都假设 $\prec_i$ 代表 $>$ 关系，并且用符号 $>$ 代替符号 $\prec_i$.

定义 1.　k-支配(k-dominate)[2].

称一个点 p_i 在空间 S 上 k-支配另外一个点 p_j，当且仅当 $\exists S'\subseteq S,|S'|=k$，使得 $\forall s_r\in S'$，有 $p_i.s_r\geq p_j.s_r$，并且 $\exists s_t\in S'$ 使得 $p_i.s_t>p_j.s_t$.

定义 2.　k-支配 skyline(k-dominant skyline)[2].

一个点 p_i 成为空间 S 上关于数据集 D 的一个 k-支配的 skyline 点，当且仅当不存在任何点 p_j，满足 $p_i\in D,p_j\neq p_i$，并且使得 p_j k-支配 p_i. 用 $DSP(k,D,S)$ 代表空间 S 上关于数据集 D 的 k-支配 skyline 点集.

显然，当 $k=d$ 时，k-支配关系等同于传统 skyline 支配关系.

定理 1.　对于一个给定空间 S 上的任意数据集 D，其 k-支配 skyline 数据点的数量随着 k 的取值变小而呈单调递减[2]，即

$$|DSP(k,D,S)|\leq|DSP(k+1,D,S)|.$$

为了证明上面定理，我们先证两个引理.

引理 1.　对于任意两个点 p_i 和 p_j，如果点 p_i $(k+1)$维-支配点 p_j，则点 p_i 必定 k 维-支配点 p_j.

证明.　根据 k-支配的定义，若点 p_i $(k+1)$维-支配点 p_j，则存在 $S'=\{s_1,s_2,\cdots,s_{k+1}\}\subseteq S$ 使得对于任意 $s_t\in S'$，有 $p_i.s_t\geq p_j.s_t$，并且至少存在一个 $s_r\in S'$ 满足 $p_i.s_r>p_j.s_r$. 显然，取 $s_r\in S'$，并任取另外 $k-1$ 个 $s_t\in S'$ 组成 k 个维度，则在这 k 个维上 $p_i.s_t\geq p_j.s_t$ 且 $p_i.s_r>p_j.s_r$. 因此 p_i k-维支配点 p_j. 引理得证.　　证毕.

引理 2.　如果一个点是 k 维-支配的 skyline 点，则它也是$(k+1)$维-支配的 skyline 点. 即

如果 $p_i\in DSP(k,D,S)$，则 $p_i\in DSP(k+1,D,S)$.

证明.　用反证法. 假设存在一个点 $p_i\in D$，满足 $p_i\in DSP(k,D,S)$，但 $p_i\notin DSP(k+1,D,S)$，则必然存在另外一个点 $p_j\in D$，使得点 p_j $(k+1)$维-支配点 p_i. 又由引理 1 我们可以得出，点 p_j k 维-支配点 p_i，由此推出点 $p_i\notin DSP(k,D,S)$，与假设矛盾. 定理得证.　　证毕.

由上述两个引理,定理 1 得证.

根据定理 1,可以得出对于一个数据集,只要给定足够小的 k,则能够将最终 k-支配 skyline 点的数量控制在一个有效的范围里,以便于向用户展现结果,这保证了 k-支配 skyline 查询的有效性.

k-支配 skyline 具有一个特殊的性质:对于一个数据集中的数据点,它们之间可能形成循环的 k-支配关系.表 1 的例子说明了这种循环的 k-支配关系.

表 1 循环 k-支配的一个例子

点	S_1	S_2	S_3	S_4
a	4	4	4	4
b	8	3	3	3
c	7	8	2	2
d	6	7	8	1

当 $k=3$ 时,数据点 a k-支配点 b,点 b k-支配点 c,点 c k-支配点 d,而点 d 则反过来 k-支配点 a,4 个点之间形成了循环的 k-支配关系.这种循环的支配关系给 k-支配 skyline 的计算带来了困难,传统的 skyline 算法不适用于 k-支配 skyline 的计算.

3 相关研究

现有的 k-支配 skyline 查询算法主要有 3 种:One-Scan 算法、Two-Scan 算法还有 Sorted Retrieval 算法[2],下面简要介绍这 3 种算法,详细细节可参看文献[2].

One-Scan 算法使用与传统的循环嵌套算法(BNL)相似的思想,它主要基于以下两个特性.

它是在假设一个数据点 $p\in D$ 不是一个 k-支配的 skyline 点的前题下.

性质 1. 数据集 D 中必定存在一个传统的 skyline 点 q,满足点 q k-支配点 p.

性质 2. 数据点 p 有可能不被 D 中的任一 k-支配 skyline 点所 k-支配.

根据性质 2,要判定数据集 D 中的一个点 p 是否被数据集 k-支配,仅仅用点 p 与 D 中所有 k-支配 skyline 点去比较是不够的.而根据性质 1,只要再将 p 与数据集 D 中所有传统 skyline 点进行比较就能判定它是否是一个 k-支配的 skyline 点.

基于以上两个性质,One-Scan 算法通过计算出数据集 D 的传统 skyline 点集,并用它们来淘汰 D 中的非 k-支配数据点,从而求解出结果.一般在算法的进行过程中,会维护两个中间数据集:

(1)候选 k-支配 skyline 数据点集 R.

(2)候选非 k-支配 skyline 点的传统 skyline 数据点集 T.

对每个新处理的数据点,首先与 T 中的点进行比较,如果数据点被 T 中的点所支配则直接丢弃.如果不被支配则再与 R 中的点比较,判断其是否是候选 k-支配 skyline 点,如果是则将其插入 R 中,否则插入 T 中.

虽然 One-Scan 算法不需要任何的预处理,并且只需要对数据集进行一次扫描,然而为此算法必须在整个处理过程中保留所有的传统 skyline 数据点,在高维的数据集,尤其是高维反相关的数据集中,传统 skyline 数据集几乎接近于原数据集的大小.在这种最坏情况下,算法需要多于原数据集一倍的额外空间来记录两个中间数据集.与此同时,算法的时间开销也非常大,经过分析其计算时间与数据集中的传统 skyline 点数成正比.

Two-Scan 算法为了降低计算对于空间的要求,通过对输入数据集的两次扫描,算法可以不需要维护中间数据集 T.在第一次对数据集的扫描中,循环地将 D 中的每一个点 p 与候选 k-支配 skyline 数据集 R 中的每一个点进行比较,如果 p 被 R 中任意点 k-支配,则丢弃 p,否则将其插入 R 中,并且将 R 中被 p k-支配的点删除掉.在第一次扫描处理结束后,根据性质 2,R 中包含了非 k-支配 skyline 的点.因此算法通过第二次扫描去除 R 中的错误结果,将 R 中的点与 D 中的每个点进行比较.如果 R 中的一个点 p 被 D 中的任意点 k-支配,则将 p 丢弃.经过第二轮扫描后得到的数据集 R 就是 D 中的 k-支配 skyline 点集.

Two-Scan 算法的效率主要取决于在第一次扫描过程中,保存在 R 中的候选 k-支配 skyline 数据点能否有效地淘汰掉大部分不属于结果的数据点.如果在第一次扫描过后,R 中的非 k-支配 skyline 点数量很少,则第二次扫描的代价很小,从而整个算法的性能会比较好.但是,由于存在着循环 k-支配的关系,算法可能在第一次扫描过后在数据集 R 中保留了大量非 k-支配 skyline 数据点,使得第二轮扫描的计算开销会很大,在最坏情况下算法的总时间复杂度会超过 One-Scan 算法,并且空间开销上也体现不出优势.

对于大规模的数据集,计算时间可能很长,用户往往希望算法能够在计算过程中快速地返回部分结果,实现渐进输出.前两种算法都必须在计算结束后

才能返回查询结果，因此不具备渐进性.

Sorted Retrieval 算法可以实现结果的渐进输出，它的思想来源于文献[3]. 首先算法将数据集 D 中的点根据每一个维度 $s_i \in S$ 分别进行排序(降序排序)，将每个维度的排序结果存储在 $|S|$ 个数组中. 在具体的实现过程中，为了节省存储空间，一般用指针代替实际存储的数据点. 算法在处理过程中维护一个数据集 T，一开始将 T 初始化为 D. 不同于前两个算法顺序地对数据集 D 进行扫描，Sorted Retrieval 算法反复地用一个选择函数在这 $|S|$ 个数组中选择一个数组，对该数组上未被处理的具有最小值的一堆数据点进行计算，将被 k-支配的数据点从 T 中丢弃，同时将被确定为属于 k-支配 skyline 的数据点输出. 这个过程反复循环，直到数据集 T 为空. 在算法的处理过程中，一个数据点 p 只要满足以下两个条件，就可以被判定为是 k-支配的 skyline 点：(1) p 没有被已经处理过的数据点所 k-支配(即 p 仍然留在数据集 T 中)；(2) p 已经被算法处理过 $d-k+1$ 次. 根据以上条件，算法不需要完成全部计算就可以提前输出部分结果.

由于 Sorted Retrieval 算法根据各个维度对数据进行了预排序，因此在计算过程中具有支配能力的数据点有较高的概率被提前处理并用于快速淘汰其它不属于结果的数据点. 同时算法具备了渐进性. 然而，由于 Sorted Retrieval 算法需要记录每个维度的预排序结果数组，在具体实现中虽然用指针可以避免冗余存储数据点，但是 $|D|$ 个指针数组也使得算法需要的额外空间开销很大. 同时，预排序的开销并不一定能保证算法在性能上比另外的算法有更好的表现. 尤其在反相关的数据集中，大部分数据点只能在算法的结束阶段才被输出，这时算法在计算效率上并不比 Two-Scan 算法高，再加上预排序的开销，性能反而更差了. 这种不理想效果很大程度上是因为每个维度的预排序结果数组只保留了单个维度的信息，而没有根据 k 值对数据点进行一个综合的考虑.

4　基于索引的高效 k-支配 Skyline 算法

4.1　算法的基本思想

我们的算法从时间效率、空间开销和渐进输出性 3 个方面进行考虑.

(1) 时间效率的改进

为了提高算法的时间效率，应该尽可能地减少点与点之间的比较次数.

在对数据集 D 进行 k-支配 skyline 查询的过程中，如果算法在判定一个数据点 $p \in D$ 是否属于 k-支配 skyline 数据集时，能够将 p 优先与 D 中具有较强 k-支配能力的数据点进行比较，则对于大部分非 k-支配 skyline 数据点，算法可以快速地将它们淘汰掉，从而节省了大量比较，提高了效率. 为了实现这一优化目标，算法需要建立一个关于数据集 D 中各个数据点 k-支配能力的索引.

(2) 提高渐进输出性

一个渐进性良好的算法必须能够在计算初期快速返回部分结果. 为了尽快输出结果，在对数据点的处理过程中算法应该优先选择那些有较高概率属于结果数据集的点进行处理.

在算法对数据点进行处理时，如果能够在未被处理的数据点中优先选择具有较大可能成为 k-支配 skyline 的点，则可以更快地找到大部分目标数据点，从而减少算法需要维护的中间数据集的大小，并且能够较快地返回部分计算结果，保证算法良好的渐进性. 为此，算法还需要建立一个关于数据集 D 中各个数据点成为 k-支配 skyline 点能力的索引.

通过建立以上的两个索引表，算法在处理过程可以根据第 2 个索引来选择要处理的数据点，再将当前处理的数据点根据第 1 个索引的顺序与其它数据点进行比较，这样算法就能够快速识别并返回真正的 k-支配 skyline 点，同时快速淘汰其它的非结果数据点，实现高效率和良好渐进性的计算.

(3) 减少空间开销

空间开销的优化目标是减少计算过程中需要的额外空间，并且保持开销大小的稳定.

由于我们只为数据集建立两个索引，比起需要建立 $|S|$ 个预排序数组的 Sorted Retrieval 算法，能够大大降低建立索引的空间开销. 同时改进算法使得计算过程中不需要保留中间结果，这样与需要暂存中间数据集的 One-Scan 和 Two-Scan 算法相比不仅空间开销减少了，而且开销的大小也保持稳定.

4.2　算法的具体设计

1) 建立两个索引表：Ability 表和 Possibility 表

根据优化的基本思想，我们要为数据集建立索引，首先要对一个数据点 $p \in D$，定义它 k-支配其它数据点的能力和成为 k-支配 skyline 数据点可能性的形式描述.

这里假设 D 所在的空间 S 上每个维度都具有相同的取值范围，并且取值都不为负数(实际应用中可

以通过对数据集的规范化预处理来满足这一条件).

(1)根据 k-支配 skyline 的定义,一个数据点只要在任意 k 个维度上取值优于另一个数据点,则 k-支配的关系成立.如果一个点在 k 个维上取值很好,则它 k-支配其它点的能力就会比较强.因此,一个点的 k-支配能力直观上可以用它取值最优的 k 个维上的数值之和来表示.

对于一个 D 中的数据点 $p\in D$,关于参数 k,它的 k-支配能力用 $bestKDim(p,k)$ 表示:

$$bestKDim(p,k)=\sum_{i=1}^{k}maxdim(p,i) \quad (1)$$

其中 $maxdim(p,i)$ 为数据点 p 所有维上取值第 i 大的值.

式(1)直接对数据点取值最优的 k 个维求和,对于各个维上数值差别很大的数据,会造成数据点的某些维在总和中具有过高的权重,因此直接用 k 个维之和不能很好地反应数据点的 k-支配能力.考虑下面的例子,表 2 中分别是数据点 p 和 q 的两个最优维度($k=2$).

表 2 p 和 q 取值最优的两个维度的数值

p	q
5	1
5	10

如果使用式(1)来表示数据点的 k-支配能力,则点 q 的 k-支配能力强于点 p.然而实际上,点 p 比点 q 具有更大可能支配其它数据点.如图 1 所示:在各自取值最优的 2 维空间上点 p 比点 q 具有更大的 k-支配面积.

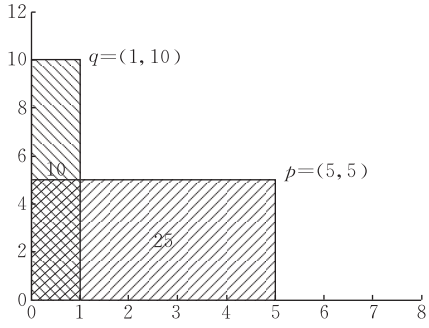


图 1 p 和 q 的 k-支配面积

图 1 的例子同样可以扩展到 k 大于 2 的情况,因此为了更客观地反应数据点的 k-支配能力,应该用数据点最优的 k 个维的乘积来表示,将 $bestKDim(p,k)$ 修改为

$$bestKDim(p,k)=\prod_{i=1}^{k}maxdim(p,i) \quad (2)$$

考虑到当 k 的取值较大时,式(2)在算法的实际实现中容易导致溢出.为了解决这一问题,我们可以引入对数函数 $\ln(x)$,注意到有(假设对于任意的 p 和 i,$maxdim(p,i)>0$)

$$\ln\left(\prod_{i=1}^{k}maxdim(p,i)\right)=\sum_{i=1}^{k}\ln(maxdim(p,i)) \quad (3)$$

再考虑到取值为 0 的情况,在算法中对每个 $maxdim(p,i)$ 先加 1 再求相应的对数函数值(根据前面的假设,每个维度的取值均非负).对于一个 D 中的数据点 $p\in D$,关于参数 k,它的 k-支配能力最后表示为

$$bestKDim(p,k)=\sum_{i=1}^{k}\ln(maxdim(p,i)+1) \quad (4)$$

即 $bestKDim(p,k)$ 为点 p 所有维中取值最大的 k 个维上数值加 1 关于对数函数 $\ln(x)$ 的值之和.

表 3 中分别是数据点 p 和 q 的两个最优维度用式(1)与使用式(4)做度量的数值.可以看出,在直接求和的情况下 q 优于 p,而在使用对数函数之后,p 优于 q.显然,从图 1 直观上我们知道 p 在大部分情况下比 q 具有更强的 k-支配能力,在直接求和的情况下,q 只要在一个维上有非常好的取值,就可以弥补另外一个维上的缺陷,这使得总和不能客观反映各个维的情况.而通过使用 $\ln(x)$ 之后的总和则可以防止单个维度权重过高,从而更全面地反映数据点的 k-支配能力.

表 3 直接求和与使用对数函数的能力比较

	直接求和	使用对数函数 $\ln(x)$
p	10	3.21888
q	11	2.30259

根据式(4)对 $bestKDim$ 的定义,为数据集中的每一个数据点 p 计算其 $bestKDim(p,k)$ 值,然后根据该值由高到低建立所有数据点 k-支配能力的索引 Ability 表.

(2)一个数据点要成为 k-支配 skyline,必须保证不被数据集中任意其它数据点所 k-支配.如果一个点在最差的 k 个维上的取值仍然比较好,则它很难被其它点所 k-支配,有很大的可能成为 k-支配 skyline 点.因此,一个数据点成为 k-支配 skyline 点的可能性可以直观地用它取值最差的 k 个维上的数值之和来表示.

与(1)同理,对于一个 D 中的数据点 $p\in D$,关于参数 k,它成为 k-支配 skyline 数据点的能力用 $worstKDim(p,k)$ 表示:

$$worstKDim(p,k)=\sum_{i=1}^{k}\ln(mindim(p,i)+1) \tag{5}$$

其中 $mindim(p,i)$为数据点 p 所有维上取值第 i 小的值，即 $worstKDim(p,k)$为点 p 所有维度中取值最小的 k 个维上数值加 1 关于对数函数 $\ln(x)$的值求和.

根据 $worstKDim$ 的定义，为数据集中的每个数据点 p 计算其 $worstKDim(p,k)$值，然后根据该值由高到低建立所有数据点成为 k-支配 skyline 可能性的索引 Possibility 表.

2）根据索引进行计算

在建立完数据集的两个索引之后，算法根据索引进行计算. 按照 Possibility 表的索引顺序选择数据集中的点进行处理. 对每个处理的数据点 p，将它按照 Ability 表的索引顺序与其它数据点进行比较，一旦 p 在计算过程中被其它点 k-支配，则立即将其淘汰. 若点 p 与 Ability 表中的所有点比较后仍未被淘汰，则 p 成为 k-支配 skyline 点，可以立即输出. 接着算法再从 Possibility 表中取下一个点进行判断，如此迭代直到所有点都被处理过则算法结束.

3）计算过程中的进一步优化

对于数据集中每一个 k-支配 skyline 点，算法必须将它与所有点都进行比较. 这是因为只有与所有的数据点都进行比较之后，算法才能确定一个点属于 k-支配 skyline. 然而，在 Ability 表中排在后面的那些点其 k-支配能力很弱，对于在 Possibility 表中排名很前的数据点，将它们进行比较是多余的，算法应该尽可能减少这种无意义的比较.

实际上在计算过程中依靠索引不仅可以提高对非结果数据点的淘汰效率，而且在判断一个 k-支配 skyline 点时也可以减少不必要的比较. 为此在算法中我们需要用到一个定理以实现进一步的优化，下面我们提出并证明这个定理.

定理 2.　对于数据集 D 中的两个数据点 p 和 q，关于参数 k，如果满足 $worstKDim(p,k)\geq bestKDim(q,k)$，则点 p 不会被点 q 所 k-支配.

证明.　用反证法. 假设数据集 D 中存在两个数据点 p 和 q，关于参数 k，满足 $worstKDim(p,k)\geq bestKDim(q,k)$，又点 q k-支配点 p.

由于点 q k-支配点 p，根据 k-支配的定义，则存在 $S'=\{s_1,s_2,\cdots,s_k\}\subseteq S$，满足 $\forall s_t\in S'$，有 $p.s_t\leq q.s_t$，且存在 $s_t\in S'$，使得 $p.s_t<q.s_t$. 由于对数函数 $\ln(x)$ 是 $(0,+\infty)$ 上的单调递增函数，因此，$\sum_{s_t\in S'}\ln(p.s_t+1)<\sum_{s_t\in S'}\ln(q.s_t+1)$.

同时，根据 $worstKDim(p,k)$的定义，有 $worstKDim(p,k)\leq\sum_{s_t\in S'}\ln(p.s_t+1)$，根据 $bestKDim(q,k)$的定义有 $\sum_{s_t\in S'}\ln(q.s_t+1)\leq bestKDim(q,k)$，由此可以推出

$$worstKDim(p,k)\leq\sum_{s_t\in S'}\ln(p.s_t+1)<\sum_{s_t\in S'}\ln(q.s_t+1)\leq bestKDim(q,k),$$

这与前面的假设 $worstKDim(p,k)\geq bestKDim(q,k)$矛盾，结论得证.　　证毕.

根据以上定理，在算法的计算过程中，如果正在处理的数据点 p 的 $worstKDim$ 值大于或等于 Ability 表中未与 p 比较的数据点的 $bestKDim$ 值，则可以提前判定点 p 是 k-支配 skyline 点，不需要继续与 Ability 表中的剩余点进行比较.

4）算法的伪代码描述

算法具体实现的伪代码描述如算法 1.

算法 1.　*Index-Based*(D,S,k).

```
;算法输入包括 3 个参数，D 代表查询的数据集，S 代表数据集所在的空间，k 代表查询 k-支配 skyline.
;算法输出 D 中的 k-支配 skyline 数据集 R.
1. for every point p∈D do
2.    compute the bestKDim(p,k) and worstKDim(p,k) value of p
3.    initialize the marker for p, isDominated=false
4. sort D in non-ascending order of each point's worstKDim(p,k) value and store the pointer to each point in array Possibility[1…|D|]
5. sort D in non-ascending order of each point's bestKDim(p,k) value and store the pointer to each point in array Ability[1…|D|]
6. initialize the set of k-dominant skyline points R=∅
7. initialize the cursor for Possibility, c_worst=1
8. while (c_worst≤|D|) do
9.    let p be the point pointed by Possibility[c_worst]
10.   if (p.isDominanted=false) then
11.      initialize the cursor for Ability, c_best=1
12.      while (c_best≤|D|) do
13.         let q be the point pointed by Ability[c_best]
14.         if (worstKDim(p,k)≥bestKDim(q,k)) then
15.            break out of the inner while-loop
16.         if (p k-dominates q) then
17.            q.isDominanted=true
18.         if (q k-dominates p) then
19.            p.isDominated=true
```

```
20.            break out of the inner while-loop
21.          c_best = c_best + 1
22.        if (p.isDominanted = false) then
23.          insert p into R
24.    c_worst = c_worst + 1
25. return R
```

在算法的开始阶段,首先为每个数据集 D 中的数据点 $p \in D$,根据参数 k 计算其 $worstKDim(p,k)$ 和 $bestKDim(p,k)$ 值,并为每个点初始化一个标识变量 $isDominated$ = false,用来记录该数据点是否已经被其它数据点所 k-支配(步 2～3). 接着根据 D 中数据点的 $worstKDim(p,k)$ 值,按照单调递增顺序来建立关于 D 中数据点成为 k-支配 skyline 点能力的索引表 $Possibility[1\cdots|D|]$. 再根据所有点的 $bestKDim(p,k)$ 值,按照单调递增顺序建立关于数据点 k-支配能力的索引表 $Ability[1\cdots|D|]$(步 4～5).

建立完索引后,算法按照 $Possibility[1\cdots|D|]$ 表的索引顺序选择要处理的数据点(步 8～9). 对每个处理的数据点 p,按照 $Ability[1\cdots|D|]$ 表的索引顺序选择数据点 q 跟 p 进行比较,并根据比较的结果更新点 p 和点 q 的标识变量 $isDominated$(步 11～20).

注意到在处理每个数据点 p 之前,算法都先判断 p 的标识 $isDominated$,如果为 true 则可以直接忽略(步 10). 因为点 p 有可能在被 $Possibility[1\cdots|D|]$ 表索引之前就被 $Ability[1\cdots|D|]$ 表索引用来与被处理的数据点进行比较,算法通过 $isDominated$ 值便可以尽可能避免重复的比较. 此外,在处理一个数据点 p 时,如果对于 $Ability[1\cdots|D|]$ 中剩余的任意数据点 q,$worstKDim(p,k)$ 值不小于 $bestKDim(q,k)$ 值,则根据定理 2,p 可以被确认为 k-支配 skyline 点,因此不需要再继续比较下去(步 14～15). 这一优化步骤可以进一步避免计算中多余的比较,提高算法性能.

在每一轮循环结束后,算法可以直接判定该轮循环中所处理的数据点是否是 k-支配的 skyline 点,因此能够在每一轮循环结束后立即返回已经计算的结果(步 22～23),从而具有比 Sorted Retrieval 算法更好的渐进性.

5 分析与实验

5.1 空间开销和渐进性分析

(1) 空间开销

设数据集 D 的维数为 d,包含的点数为 n,设算法的参数为 k.

对于给定的数据集 D 和参数 k,本文提出的 Index-Based 算法的空间开销是稳定的. 它需要建立两个索引,所需要的空间大小与数据集的维数 d 无关,只受数据集点数 n 的影响,建立索引需要的空间开销为 $O(2n)$. 此外,算法在执行过程中没有中间结果需要存储,因此算法总的空间复杂度也为 $O(2n)$.

在高维空间中,随着维数 d 的增加,Index-Based 算法相对于其它 3 种算法在空间复杂度上的优势会更加明显,显示出良好的可扩展性. 并且在输入数据集大小超过可用内存空间的情况下,这一优势显得更为重要,能避免算法的大量 IO 操作.

(2) 渐进输出性

Index-Based 算法根据索引优先选择具有最大可能成为结果的数据点进行处理,因此在算法的执行初期就能返回大部分的执行结果,具有良好的渐进性.

表 4 对 4 种算法在空间开销和渐进性这两个方面进行了总结.

表 4 4 种算法的性能对比

	空间稳定性	最差空间复杂度	可扩展性	渐进性
One-Scan	不稳定	$O(dn)$	差	没有
Two-Scan	不稳定	$O(dn)$	差	没有
Sort Retrieval	稳定	$O(dn)$	差	差
Index-Based	稳定	O(2n)	好	好

5.2 时间效率实验

为了评估算法的时间效率,将本文提出的算法与现有的 3 种 k-支配 skyline 算法进行对比. 对 4 种算法均采用相同的编程语言和运行环境(表 5).

表 5 测试环境和数据集

编程语言	运行环境		测试数据	
	CPU	内存	合成数据集	真实数据集
C++,用 VC6.0 开发和编译	Intel Pentium 4 2.4GHz	1GB	由数据生成器生成测试数据	NBA 球员统计数据集

(1) 实验参数说明

实验中合成数据集所使用的数据生成器由香港中文大学提供,合成数据集需要用到如表 6 的参数.

表 6 实验的参数

参数	说明
d	测试数据集的维数
Size	测试数据集的数据点数
Distribution	测试数据集的分布
k	k-支配 Skyline 计算的定义参数 k

用于 skyline 算法测试的数据集一般有以下 3 种分布:正相关(correlated)、独立(independent)和反

相关(anti-correlated). 在正相关的数据集中，各个维度的数值大小之间呈正相关的关系. 在独立分布的数据集中，各个维度在取值上互不相关. 在反相关数据集中，由于各维度的数值之间是反相关的关系，任意一个数据点在某个维度上数值很高，则会在另外一个维度上数值较低.

在下面的实验中，将对 4 种算法在表 6 中各个参数变化下的性能进行比较. 在默认的情况下，各个参数设置为 $d=15$，$Size=100000$，$Distribution=Independent$，$k=11$. 为了便于描述，对各个算法及其扩展算法，下文中将使用如下简称：OSA 代表 One-Scan 算法，TSA 代表 Two-Scan 算法，SRA 代表 Sorted Retrieval 算法，IBA 代表 Index Based 算法.

(2) 合成数据集测试结果

图 2 是 4 种算法在参数 k 变化下的性能比较，从图中可以看出，OSA 的性能总体上是最差的. TSA 在参数 k 较小的情况下，性能比 SRA 更优，这是由于 k 越小则数据点形成 k-支配关系的可能性越大，在第一轮扫描中 TSA 能够淘汰掉大部分的数据点. 但是随着 k 的增长，TSA 性能下降很快，在反相关数据集测试中当 $k=14$ 时 TSA 的计算时间甚至超过了 OSA. SRA 的性能相对比较稳定. 在 4 种算法中，IBA 性能稳定，并且计算速度始终保持最快.

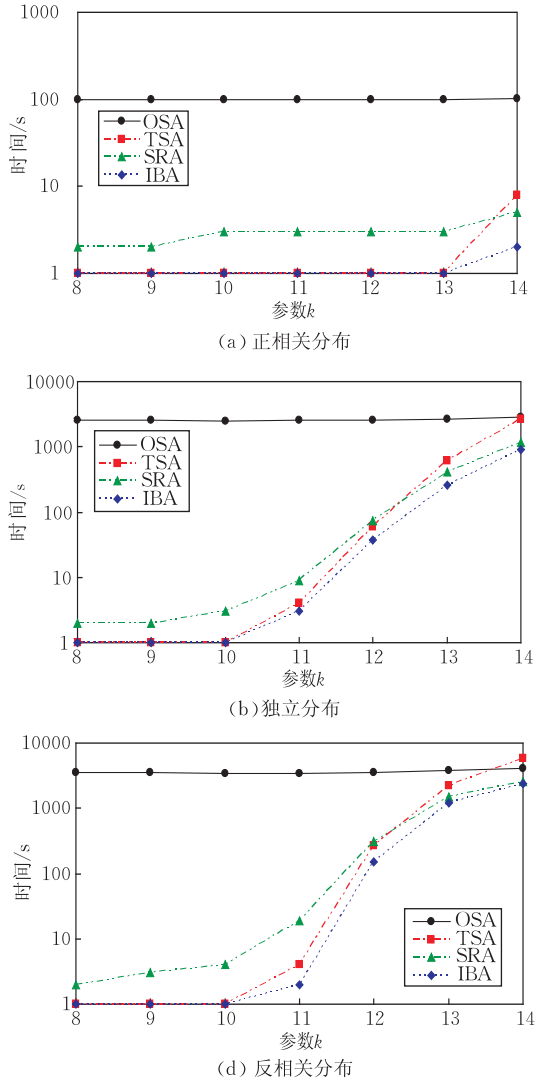


图 2　在参数 k 变化下的算法性能

图 3 是关于数据集维数变化的测试结果(其中 10D6K 表示 $d=10$，$k=6$). OSA 的性能虽然受维数增加的影响不大，但总体上仍是最差的. TSA 受维数变化的影响最大，在维数 $d=20$ 的独立分布和反相关分布数据集上 TSA 的时间开销甚至是两倍于 OSA. SRA 和 IBA 算法表现稳定，而 IBA 比 SRA 受维数增加的影响更小，在所有算法中是最优的.

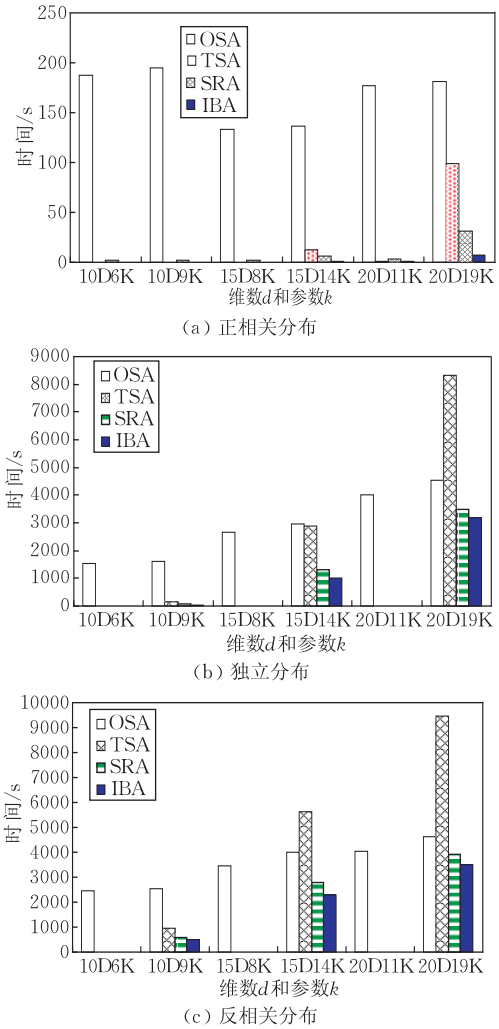
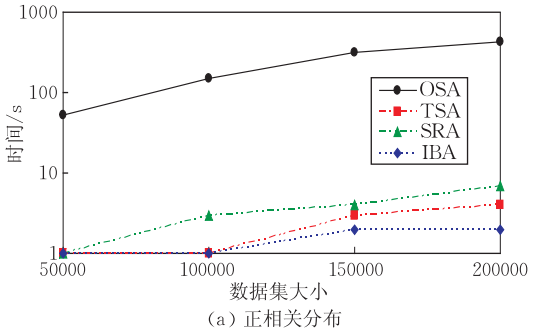
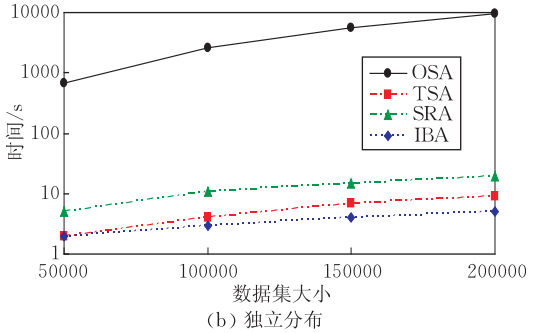


图 3　在维数 d 变化下的算法性能

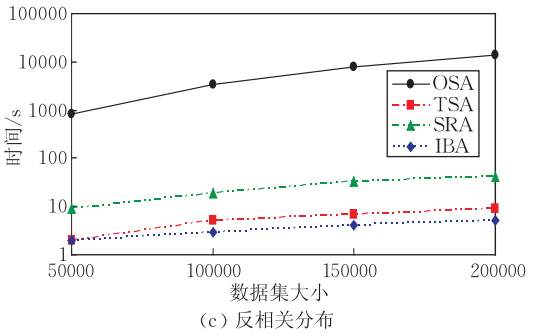
图 4 是在数据集大小 $Size$ 变化下的算法性能比较，随着数据集规模的增加，OSA 的性能严重下降. 另外 3 种算法的表现是 IBA 最好、TSA 其次，然后是 SRA. 这表明 IBA 算法比其它 3 种算法有更好的可扩展性.



(a) 正相关分布



(b) 独立分布



(c) 反相关分布

图 4 在数据集大小 *Size* 变化下的算法性能

(3) 真实数据集测试结果

NBA 球员统计数据集[①]包含了 NBA 所有球员在各个赛季表现的统计,包括得分、篮板、抢断等等共 17 项技术统计,整个数据集共 17801 个数据点. 对数据集的测试主要测试 k-支配算法在不同的 k 值下的性能表现.

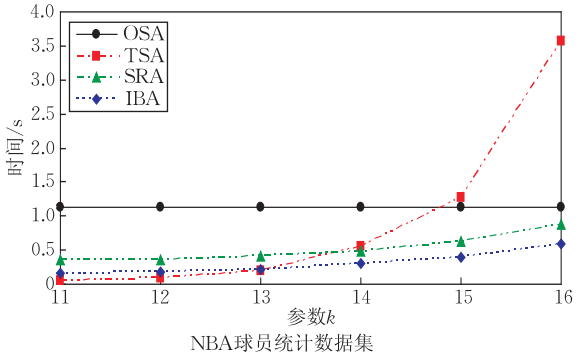


图 5 NBA 球员统计数据集的测试结果

从图 5 的实验结果看,在 NBA 统计数据集中,由于数据量不是很大,在参数 k 小于 12 的时候,k-支配 skyline 点非常少,不需要预计算的 TSA 速度是最快的. 不过随着 k 的增加,TSA 的时间开销迅速增加,很快超过了平均性能最差的 OSA. 而 IBA 性能非常稳定,在总体表现上明显优于另外 3 种算法.

6 结 论

k-支配 skyline 通过降低对于"支配"的定义,使得挖掘出来的 k-支配 skyline 数据集具有更重要的意义和作用. 然而由于 k-支配 skyline 的特殊性质,传统的算法无法适应其计算要求,而现有根据 k-支配 skyline 定义提出的算法在时间效率、空间复杂度和渐进性上都存在不足.

本文提出一种基于索引的 k-支配 skyline 算法,通过建立两个有效的索引,算法可以尽量减少数据点之间的比较次数,提高了时间效率;尽可能快地返回查询结果,具有良好的渐进性;只使用两个索引,并且不需要保存中间结果,节省了计算的空间开销. 实验证明本文提出的算法比现有的 3 种算法有更好的性能.

k-支配 skyline 研究方面还有很多改进和扩展的空间,如:能否通过建立更加复杂有效的索引,使得算法能够具有更好的效率和可扩展性. 与此同时,子空间 skyline 的挖掘是 skyline 的另一个研究热点,而 k-支配 skyline 算法的特殊性质使得其在子空间 skyline 计算上具有重要意义,如何将两者进行结合是一个值得深入探讨的问题.

参 考 文 献

[1] Stephan Börzsönyi, Donald Kossmann, Konrad Stocker. The skyline Operator//Proceedings of the 17th International Conference on Data Engineering. Heidelberg, Germany, 2001: 421-430

[2] Chan Chee-Yong, Jagadish H V, Tan Kian-Lee, Tung Anthony K H, Zhang Zhen-Jie. Finding k-dominant Skylines in high dimensional space//Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data. Chicago, Illinois, USA, 2006: 503-514

[3] Ronald Fagin, Amnon Lotem, Moni Naor. Optimal Aggregation Algorithms for Middleware. Journal of Computer and System Sciences, 2001, 66(4): 614-656

① NBA Statics Database. http://www.basketballreference.com/

[4] Jan Chomicki, Parke Godfrey, Jarek Gryz, Dongming Liang. Skyline with Presorting//Proceedings of the 19th International Conference on Data Engineering. Bangalore, India, 2003: 717-816

[5] Tan Kian-Lee, Eng Pin-Kwang, Ooi Beng Chin. Efficient progressive skyline computation//Proceedings of the 27th International Conference on Very Large Data Bases. Roma, Italy, 2001: 301-310

[6] Donald Kossmann, Frank Ramsak, Steffen Rost. Shooting stars in the sky: An online algorithm for skyline queries//Proceedings of the 28th International Conference on Very Large Data Bases. Hong Kong, China, 2002: 275-286

[7] Papadias Dimitris, Tao Yufei, Fu Greg, Seeger Bernhard. An optimal and progressive algorithm for skyline queries//Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data. San Diego, California, USA, 2003: 467-478

[8] Yiu Man Lung, Mamoulis Nikos. Efficient processing of top-k dominating queries on MultiDimensional data//Proceedings of the 33rd International Conference on Very Large Data Bases. Vienna, Austria, 2007: 483-494

[9] Chan Chee-Yong, Jagadish H V, Tung Anthony K H, Zhang Zhen-Jie. On high dimensional Skylines//Proceedings of the 10th International Conference on Extending Database Technology. Munich, Germany, 2006: 478-495

YIN Jian, born in 1968, professor, Ph. D. supervisor. His major research interests include database, data mining, artificial intelligence.

YAO Shu-Yu, born in 1983, master. His research interests include database and data mining.

XUE Shao-E, born in 1983, master. His research interests include database and data mining.

YANG Wen-Xin, born in 1984, master. His research interests include database and data mining.

LIU Yu-Bao, born in 1975, associate professor. His research interests include database and data mining.

Background

Skyline query, which was first proposed by Borzsonyi et al. in 2001[1], extracts interesting information for decision-making applications and has recently received a lot of attentions in the fields of database and data mining. A point p is said to dominate another point q if it is better than or equal to q in all directions and beats q in at least one. Skyline query returns the skyline points, which are not dominated by any other one. Skyline query is useful for decision-making support, for instance, we can apply skyline query to help us sieve out the commodities that worth our consideration, and then select the merchandise that we want in the skyline query result.

In the past few years, researcher in this field focus on two aspects, one is how to efficiently find the skyline points in massive data, some efficient algorithms are proposed, including Block Nested Loop, Divide-and-Conquer, Bitmap, Index, Sort Filter Skyline, Nearest Neighbor, and Branch-and-Bound Search. The other one is to design useful form of skyline query to support decision-making. For example, 1) subspace skyline query is applied in subspaces constituted by different dimensions; 2) partial order relationship between different attributes are defined; 3) The concept of k-dominate skyline is proposed to address the issue that caused by large scale high-dimensional data.

k-dominate skyline query, which was proposed by Chee-Yong Chan et al in SIGMOD'06 is one of the most important research direction in skyline query. As the dimensionality increases, the probability for points to form dominating relationship becomes lower, resulting that the number of skyline points is too big, even equals to that of the original dataset, therefore, skyline query becomes meaningless. k-dominate skyline relaxes the definition of domination to decrease the scale of skyline set and hence more useful for decision.

Unfortunately, due to the characteristics of k-dominant skyline, the traditional skyline algorithms, such as mentioned above are not applicable for k-dominant skyline query. At present, there are mainly 3 new algorithms for k-dominant skyline, One-Scan, Two-Scan, and Sorted-Retrieval, however, they are deficient in time, space, and not online. This paper proposes a new index-based algorithm, in which two indices tables are introduced, which makes this new approach an online algorithm that has the advantages in time, space.