面向高效并行 Skyline 计算的数据划分方法

赵 翔[1),3)] 商海川[2)]

[1)](国防科技大学信息系统工程重点实验室 长沙 410073)

[2)](东京大学工业科学研究院 东京 153-8505 日本)

[3)](地球空间信息技术协同创新中心 武汉 430079)

摘 要 Skyline 计算是数据管理领域长久以来的一个研究重点和热点.给定一组多维的数据点,Skyline 算子从中筛选出在所有维度上都不被其他点支配的数据点;Skyline 算子的处理过程称之为 Skyline 计算.Skyline 算子使得用户可以在较小规模的 Skyline 结果集上选择自己感兴趣的对象,而无须关心那些已经被过滤掉的对象.因此,Skyline 计算在多目标决策、数据可视化分析、用户偏好查询等方面应用广泛,典型的应用任务包括但不限于商业营销策略分析,产品能力横向评估等.随着大数据时代的到来,以及分布式网络系统的深入应用和基于云计算平台解决方案的快速发展,各类应用领域数据规模的快速增长已经成为一个关键性技术挑战,面向大规模数据集的并行 Skyline 算子应运而生,以部分解决大数据给 Skyline 计算困难;同时,并行 Skyline 计算的相关研究近年来备受学术界和工业界的广泛关注.由于缺乏关于整个数据集的全局分布信息,并行 Skyline 计算的高效处理面临着巨大的技术挑战.一般认为,并行 Skyline 处理的计算框架通常包含三个主要步骤:(1)合理划分给定的大数据集;(2)利用本地计算资源在每个数据分块上分别计算局部 Skyline;(3)合并局部 Skyline 最终形成全局 Skyline.其中,针对后两步——计算局部 Skyline 和合并局部 Skyline 的现有算法较多,相关研究相对成熟;相较而言,第一步上的相关研究工作则较少,但其效果却直接决定了整体计算的并行化程度,进而能够影响并行计算系统的整体性能.具体地,第一步需要考虑两方面的准则:(1)各个分块上的计算负载是否均衡;(2)如何减小每个分块上局部 Skyline 的基数.然而,无论采用基于随机划分还是基于网格的方法,现有算法均只能满足上述两个准则之一,不能两全其美.针对该问题,研究探索了如何利用概率模型估计 Skyline 基数的期望,该概率模型将已有研究的相关结论纳入到了一个统一的框架中.接着,据此提出了一种新的基于排列的数据划分方法,它通过简单的数据点映射即可实现负载均衡,同时生成小于现有其他方法的 Skyline 候选点集.在理论研究的坚实基础上,在大型人工和真实数据集上实验验证了所提模型和方法的有效性;换言之,在大规模实验研究中,所提方法显著提高了并行 Skyline 算子的执行效率,在绝大多数参数设定下的表现都优于现有其他同类算法.

关键词 并行 Skyline;数据划分;排列模型;可扩展性

中图法分类号 TP311 **DOI 号** 10.11897/SP.J.1016.2020.02050

Data Partitioning Method for Efficient Parallel Skyline Computation

ZHAO Xiang[1),3)] SHANG Hai-Chuan[2)]

[1)](*Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology, Changsha* 410073)

[2)](*Institute of Industrial Science, The University of Tokyo, Tokyo* 153-8505, *Japan*)

[3)](*Collaborative Innovation Center of Geospatial Technology, Wuhan* 430079)

Abstract Skyline computation has long been an important and hot research topic in the field of data management. Given a set of multi-dimensional data points, a Skyline operator selects the points that are not dominated by any other points on all dimensions; the process of processing Skyline operator is referred as Skyline computation. Skyline operator enables users to select

收稿日期:2018-08-20;在线发布日期:2019-08-20.本课题得到国家自然科学基金(61872446,U19B2024)和湖南省自然科学基金(2019JJ20024)资助.**赵 翔**,博士,副教授,中国计算机学会(CCF)会员,主要研究方向为海量数据管理与大图数据分析、知识图谱技术.E-mail:xiangzhao@nudt.edu.cn.**商海川**(通信作者),博士,特聘研究员,主要研究领域为大数据计算与区块链技术.E-mail:shang@tkl.iis.u-tokyo.ac.jp.

objects of interest from a comparatively small set of Skyline result set, without the need of care about those objects that have been filtered out. As a consequence, Skyline computation finds various applications like multi-criteria decision-making, visual data analytics, and user preference search, etc., and typical application tasks include but not limited to business marketing strategy analysis, production capability lateral assessment, etc. With the arrival of big data era, as well as the wide application of distributed networks and the rapid development of cloud computing platform-based solutions, the increase of data volume in various application domains has become a key technical challenge, and as a consequence, parallel Skyline operator for large-scale data sets was proposed, in order to partially resolve the difficulty imposed by big data; meanwhile, related research on parallel Skyline computation has received amplified attention from both academia and industry lately. Due to the lack of global distributional information regarding the overall dataset, processing of parallel Skyline computation is facing great technical challenges. Generally speaking, the computational framework of parallel Skyline processing normally comprises three major steps: (1) appropriately partition the possibly big dataset; (2) using local computing resources to evaluate local Skyline on each partition respectively; and (3) merge local Skylines into eventually a global Skyline. Among them, there are many existing algorithms for the latter two steps, i.e., compute local Skylines and merge local Skylines, and moreover, the related research is comparatively mature; on the contrary, related research on the first step is comparatively rare, although the effect of the first step directly determines the parallelism of the entire computation, and hence, the overall performance of the parallel computing system. Specifically, for the first step, there are two criteria that need to be considered: (1) whether computation workloads are balanced across partitions; (2) how to reduce local Skyline cardinalities on each partition. However, existing parallel Skyline algorithms, no matter random partitioning or grid-based method, can only satisfy only one of the criteria but not both. To address the issue, we exploit a probabilistic model to evaluate Skyline cardinality, and the probabilistic model can encapsulate relevant results from existing literature into a unified framework. Then, based on that, propose a novel permutation-based method for data partitioning which through simple data point mapping is able to achieve both load balance and generate smaller Skyline candidate sets in comparison with other existing methods. On the solid basis of theoretical study, extensive experiments on large-scale synthetic data sets and real-life data sets verify the effectiveness of the proposed model as well as the method; in other words, in comprehensive experimental study, the proposed method improves the execution efficiency of parallel Skyline operator, and it outperforms existing algorithms alike under most parameter settings.

Keywords　parallel Skyline; data partitioning; permutation model; scalability

1　引　言

Skyline 计算求解的是一个典型的多目标优化问题，相关的早期科学研究可追溯到 20 世纪 70 年代. 其问题定义如下：给定一组多维空间中的数据点 $G=\{p_1, p_2, \cdots, p_n\}$，Skyline 计算并返回所有不被其他点“支配(dominate)”的数据点，即 Skyline 点；对于多维空间中的两个数据点 p_i 和 p_j，当同时满足下述两个条件时，称 p_i 被 p_j 支配：(1) 在一个维度上，p_j 的值小于 p_i 的值[①]；(2) 在剩余其他维度上，p_j 的值不大于 p_i 的值. Skyline 的概念和帕累托最优(Pareto optimality)在本质上是相同的，因此，Skyline 算子又称帕累托算子.

图 1 展示了解释 Skyline 算子的一个经典例子. 假设去巴拿马的拿索旅游，用户想要找一个既便

① 文中认为，值越小越优.

宜又靠近海滩的旅馆;一般来说,旅馆越靠近海滩,其价格越高.因此,旅馆与海滩接近和它的价格低,这两个目标是互斥的;通常情况下,系统很难找到一个既最便宜又最靠近海滩的“最好”的旅馆,而只能退而求其次地推荐一些用户可能感兴趣的旅馆——这些旅馆在价格和距离两个方面都不(同时)比其他旅馆差.这些旅馆所对应的数据点即为 Skyline 点,如图 1 中的虚线勾勒,本例中的所有 Skyline 点包括 a、b、c、d 和 e 五个数据点.

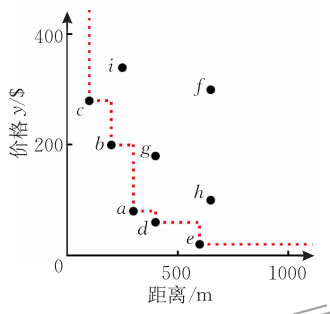


图 1 Skyline 经典示例:拿索海滩旅馆

通常 Skyline 计算采用集中式的解决方案,其上的相关研究已较为成熟,近年来也仍有一些进展[1-3].然而在大数据时代,集中式解决方案的局限性日益凸显,尤其是在效率和可扩展性方面,这种局限性变得更加明显.换言之,在面对当前的大规模在线数据分析的应用时,集中式的解决方案已经无法满足快速响应的要求.因而,并行式的 Skyline 算法应运而生[4-6],并成为近些年的研究热点.

如图 2 所示,并行 Skyline 处理的基本计算框架主要包括 3 个阶段:(1)将数据点划分到各并行计算节点;(2)各节点分别计算各自的局部 Skyline;(3)合并局部 Skyline 形成全局 Skyline.在上述框架中,数据划分起到了至关重要的作用;不合理的数据划分或直接影响多个计算节点(或数据分块①)之间的负载平衡,造成后续处理的额外开销和系统延迟.

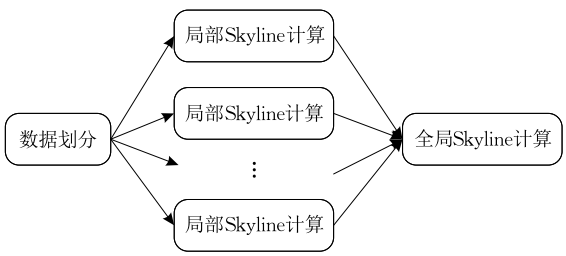


图 2 并行 Skyline 计算的基本框架

一个好的数据划分应当满足以下两个准则:(1)均衡的划分:每个工作者负责一块具有大小近似的数据划分;(2)较低的局部 Skyline 基数:每个工作者都生成一个尽可能小的局部 Skyline 点集.不幸的是,现有的并行 Skyline 算子均无法同时满足上述两个要求.具体地,并行 Skyline 计算通常采用的一种方法是基于网格(grid-based)的策略[5-9],此方法在常规数据分布上可以平衡每个划分之间的负载,但是会生成大量的局部 Skyline 点;另外,此方法是对数据分布敏感的,也就是说,当数据分布倾斜时,它仍然会产生数据划分不均衡的问题.为缓解该问题,后续研究提出了基于角度(angle-based)[10]和基于超平面投影(hyperplane-projection-based)[4]的策略以生成较少的局部 Skyline 点;但是,这两种方法却容易生成不均衡的局部数据负载.不均衡的数据负载通常意味着大差异的局部处理计算时间,这将极大地限制大型计算集群在处理海量数据时的并行化程度.

综上,现有的并行 Skyline 计算方法在高度并行和均衡负载两个准则上只能择其一满足,即基于网格的方法可以实现均衡负载但是其并行化程度较低,基于角度或超平面投影的方法可实现高度并行但是负载较不均衡.那么,能否设计一种同时满足上述两个准则——既均衡负载又高度并行的 Skyline 计算方法?下文将尝试回应上述研究问题.

具体地,针对高效并行 Skyline 计算,提出一种基于排列的数据划分方法,不仅可以有效地控制计算节点之间的负载均衡(同基于网格的方法),同时还可以生成较小的局部 Skyline 点以提高并行化程度(如基于角度和基于超平面投影的方法).总结而言,基于排列的数据划分方法具有至少如下三方面的优势:(1)每个数据分块上的局部 Skyline 基数是可理论估计的;(2)能够保证每个分块中含有近似数量的数据点,且在生成局部 Skyline 时具有近似的计算开销;(3)生成的局部 Skyline 基数远小于当前最优的其他并行 Skyline 计算方法.

综上,论文的主要贡献包括如下 3 点:

(1)针对并行 Skyline 计算问题,提出了一种用于估计 Skyline 基数的通用概率模型,将文献[11]和文献[12]中的结果纳入到一个统一的框架中;

(2)提出了一种基于排列的数据划分方法,通过简单的数据点映射即可实现负载均衡,同时生成小于现有其他方法的 Skyline 候选点集;

(3)在理论研究的基础上,在大规模人工和真实数据集上实验验证了所提模型和方法的有效性.

本文第 2 节介绍背景知识,并描述用于估计

① 文中,“划分”和“分块”表达相同的意思,可互相交换使用.

Skyline 基数的通用概率模型；第 3 节提出一种面向并行 Skyline 计算的高效数据划分方法，重点阐释其工作原理和理论保证；第 4 节是实验结果及分析；第 5 节讨论模型的扩展. 随后，在第 6 节中对相关工作进行论述；结论和下一步工作计划则放在文末的第 7 节.

2　估计 Skyline 基数的概率模型

本节从相关背景知识出发，然后提出了一个概率模型用于估计 Skyline 基数；该模型将文献[11]和文献[12]中的结果纳入到了一个统一的框架中.

2.1　背景知识

在一个 d 维数据空间中，一个数据点 $\bar{x}$ 表示为 $\bar{x}=\{x_1,x_2,\cdots,x_i,\cdots,x_d\}$，其中 x_i 是数据点 $\bar{x}$ 在第 i 维上的值，$1\leq i\leq d$. 对于两个 d 维数据点 $\bar{x}$ 和 $\bar{y}$，如果 $\bar{x}$ 在某一维 i 上的值小于 $\bar{y}$（$x_i<y_i$），且在剩余其他 $d-1$ 维上的值小于或等于 $\bar{y}$（$x_j\leq y_j$，$1\leq j\leq d, j\neq i$），则 $\bar{x}$ 支配 $\bar{y}$.

给定一个包含 n 个数据点的输入 G，Skyline 算子 $SKY(G)$ 返回 G 中一组不被其他点支配的数据点；Skyline 点集中数据点的个数称为 Skyline 基数，记作 $|SKY(G)|$，即集合中元素的个数.

在集中式环境中，最经典的 Skyline 算法是 sort-filter-skyline[13]，简称 SFS 算法. SFS 算法是一个多轮（multi-pass）算法，它在内存中设立了一个窗口来保存 Skyline 点，初始化后窗口是空的. 首先，将所有的数据点依据任一维度值进行拓扑排序，将结果记作 T_0，即第一轮待处理的数据. 假设每一轮处理一个数据集 T_i，其大小通常会超过内存的上限，因此，采取每次读取一个外存页面的方式逐次进行处理. 读取的每一个数据点 $\bar{x}$ 会和内存窗口中的每一个数据点 $\bar{s}$ 比较：若 $\bar{s}$ 支配 $\bar{x}$，则丢弃 $\bar{x}$；若 $\bar{x}$ 支配 $\bar{s}$，则丢弃 $\bar{s}$，$\bar{x}$ 还需要和窗口中剩余其他数据点比较. 若最终 $\bar{x}$ 没有被丢弃，则后续也不会再有其他 T_i 中的数据点能够支配 $\bar{x}$，$\bar{x}$ 直接是一个 Skyline 点，将 $\bar{x}$ 加入到窗口中并输出；若内存窗口不足以容纳 $\bar{x}$，则将 $\bar{x}$ 插入到 T_{i+1} 中. 当 SFS 处理到 T_i 中最后一个点时，算法结束；否则，那些在第一个数据点被写入到 T_{i+1} 中之前就存在的数据点（即 Skyline 点），可直接删除，而窗口中剩余的数据点则由下一轮处理. 之后，SFS 开始下一轮针对 T_{i+1} 中数据的处理.

在并行环境中，存在 m 个工作者（workers），如本地计算节点或亚马逊 EC2 实例①等，每个工作者记作 W_i（$1\leq i\leq m$）. G 从纵向被划分成 m 个数据分块，每个数据分块中的数据点 ρ_i 存储在 W_i 的本地存储中；且每个数据分块 ρ_i 和 ρ_j 之间没有重复的数据点，即 $\bigcup_{1\leq i\leq m}\rho_i=G$，$\rho_i\cap\rho_j=\varnothing$，$i\neq j$. 在并行 Skyline 计算中，每个工作者 W_i 利用其本地的数据 ρ_i 分别计算局部 Skyline 点集 $SKY(\rho_i)$，然后在所有局部 Skyline 点集的并集 $\bigcup_{1\leq i\leq m}SKY(\rho_i)$ 上计算出全局的 Skyline. 表 1 总结归纳了文中使用的主要符号及其含义.

表 1　主要符号与含义

符号	含义
$\bar{x}$	一个数据点
x_i	数据点 $\bar{x}$ 的第 i 维取值
d	数据空间维数
G	d 维空间中包含 n 个点的集合
$SKY(G)$	数据点集 G 上的 Skyline
$\|SKY(G)\|$	数据点集 G 上的 Skyline 基数
m	分布式集群中工作者数量
ρ_i	第 i 个工作者上的数据点集
$C_{d,n}$	度量 $\|SKY(G)\|$ 的随机变量
$\hat{C}_{d,n}$	$C_{d,n}$ 的期望
$\hat{C}_{2,n,2}$	2 维空间中，2 部划分的局部 Skyline 基数的期望之和
$\hat{C}^{\text{perm}}_{d,n,d!}$	d 维空间中，$d!$ 部划分的局部 Skyline 基数的期望之和

文中的论述将涉及两个有关数据分布的常用假设：(1) 稀疏性，数据点在各维度上不存在重复值；(2) 独立性，每个维度的取值在统计意义上是相互独立的.

用 $C_{d,n}$ 表示度量 Skyline 基数 $|SKY(G)|$ 的随机变量，$\hat{C}_{d,n}$ 是 $C_{d,n}$ 的期望值；$C_{d,n,m}$ 表示度量所有局部 Skyline 集之和 $\bigcup_{1\leq i\leq m}SKY(\rho_i)$ 大小的随机变量，$\hat{C}_{d,n,m}$ 是 $C_{d,n,m}$ 的期望值. 假设数据分布具有稀疏性，则局部 Skyline 中不存在重复值，那么，等式 $\left|\bigcup_{1\leq i\leq m}SKY(\rho_i)\right|=\sum_{1\leq i\leq m}|SKY(\rho_i)|$ 总是成立的.

在稀疏性和独立性的假设下，Godfrey 首先给出了下述关于 Skyline 基数期望的结果[11].

定理 1.　Skyline 基数的期望 $\hat{C}_{d,n}$ 等于双参数调和数

$$\hat{C}_{d,n}=H_{d-1,n},$$

① https://aws.amazon.com/ec2/

其中，$H_{k,n}$是由下式递归定义的：(1) $H_{1,n}=\sum_{i=1}^{n}\frac{1}{i}$，(2) $H_{k,n}=\sum_{i=1}^{n}\frac{H_{k-1,i}}{i}$.

此后，Chaudhuri 等人提出采用概率模型估计 Skyline 基数，并给出了如下结果[12].

定理 2. Skyline 基数的期望 $\hat{C}_{d,n}$等于

$$\hat{C}_{d,n}=n\int_0^1\int_0^1\cdots\int_0^1\left(1-\prod_{i=0}^{d}x_i\right)^{n-1}\mathrm{d}x_1\mathrm{d}x_2\cdots\mathrm{d}x_d.$$

定理 1 和定理 2 的结果从形式上看，差别较大；但是，下面我们将提出一个通用的用于估计 Skyline 技术的概率模型，据此开展理论分析，并证明定理 1 和定理 2 的结果在给定有关数据分布假设的约束条件下是等价的.

2.2 概率模型

用 Ω 表示一个抽样空间，$P(E)$是一个度量事件 $E\in\Omega$ 发生的概率. 一个随机变量 X 是一个定义在 Ω 上的函数 $X:\Omega\to\mathbb{R}^d$，其中，$\mathbb{R}^d$是一组 d 维整数向量. 分布函数 F_X定义为 $F_X(\bar{x})=P(X\leq\bar{x})$，其中，事件$\{X\leq\bar{x}\}$是针对每一个组成部分而言的(component-wise)，即

$$\{X\leq\bar{x}\}=\{X_1\leq x_1,X_2\leq x_2,\cdots,X_d\leq x_d\}=\bigcap_{k=1}^{d}\{X_k\leq x_k\}.$$

在稀疏性的假设下，Skyline 基数的估计问题可以建模成一个连续空间上的联合分布问题. 为此，这里引入一个关于变量 $\bar{x}$ 的概率密度函数 $f_X(\bar{x})=\frac{\mathrm{d}^dF_X(\bar{x})}{\mathrm{d}\bar{x}^d}$，$\bar{x}\in\mathbb{R}^d$；换言之，

$$f_{X_1,X_2,\cdots,X_d}(x_1,x_2,\cdots,x_d)=\frac{\partial^dF_{X_1,X_2,\cdots,X_d}(x_1,x_2,\cdots,x_d)}{\partial x_1\partial x_2\cdots\partial x_d},$$

其中，$x_1,x_2,\cdots,x_d\in\mathbb{R}$.

如此，上述概率模型中的变量 $\bar{x}$ 就建模了 Skyline 计算中的单个数据点，而对于任意一个给定的 $\bar{x}$，存在另外一个数据点，譬如 $\bar{x}'$，支配 $\bar{x}$ 的概率等于 $F_X(\bar{x})-P(\bigcap_{k=1}^{d}\{X_k=x_k\})$. 进一步，由于等式 $P(\bigcap_{k=1}^{d}\{X_k=x_k\})=0$ 总是成立，且仅当没有其他数据点支配 $\bar{x}$ 时，$\bar{x}$ 是一个 Skyline 点，所以，把另一个数据点支配 $\bar{x}$ 的概率定义成一个新的随机变量 $Y:\Omega\to\mathbb{R}$，则有 $Y=(1-F_X(\bar{x}))^{n-1}$. 那么，Y 的期望等于 $E(Y)=\int_{\mathbb{R}^d}(1-F_X(\bar{x}))^{n-1}F_X(\bar{x})\mathrm{d}\bar{x}$.

在稀疏性假设下，Skyline 基数的期望 $\hat{C}_{d,n}$等于 $\hat{C}_{d,n}=nE(Y)=n\int_{\mathbb{R}^d}(1-F_X(\bar{x}))^{n-1}F_X(\bar{x})\mathrm{d}\bar{x}$. 若进一步施加独立性的假设，则有

$$F_X(\bar{x})=F_{X_1}(x_1)F_{X_2}(x_2)\cdots F_{X_d}(x_d),$$

$$f_X(\bar{x})=f_{X_1}(x_1)f_{X_2}(x_2)\cdots f_{X_d}(x_d).$$

那么，综上可以得到如下的理论结果.

定理 3. 在稀疏性和独立性假设下，Skyline 基数的期望等于

$$\hat{C}_{d,n}=n\int_0^1\int_0^1\cdots\int_0^1\left(1-\prod_{i=1}^{d}x_i\right)^{n-1}\mathrm{d}x_1\mathrm{d}x_2\cdots\mathrm{d}x_d=\sum_{k=1}^{n}(-1)^{k-1}\binom{n}{k}\frac{1}{k^{d-1}}=H_{d-1,n},$$

其中，$H_{k,n}$是由下式递归定义的：$H_{1,n}=\sum_{i=1}^{n}\frac{1}{i}$，$H_{k,n}=\sum_{i=1}^{n}\frac{H_{k-1,i}}{i}$.

证明. 在稀疏性和独立性假设下，

$$\hat{C}_{d,n}=n\int_{\mathbb{R}^d}(1-F_X(\bar{x}))^{n-1}F_X(\bar{x})\mathrm{d}\bar{x}=n\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}\cdots\int_{-\infty}^{+\infty}\left(1-\prod_{i=0}^{d}F_{X_i}(x_i)\right)^{n-1}\cdot\prod_{i=0}^{d}f_{X_i}(x_i)\mathrm{d}x_1\mathrm{d}x_2\cdots\mathrm{d}x_d.$$

若定义变量 $y_i\triangleq F_{X_i}(x_i)$，则积分的上下限可分别变换为 $F_{X_i}(-\infty)=0$ 和 $F_{X_i}(+\infty)=1$，那么，

$$\hat{C}_{d,n}=n\int_0^1\int_0^1\cdots\int_0^1\left(1-\prod_{i=0}^{d}y_i\right)^{n-1}\mathrm{d}y_1\mathrm{d}y_2\cdots\mathrm{d}y_d.$$

当 $d\geq2$，$n\int_0^1\int_0^1\cdots\int_0^1\left(1-\prod_{i=0}^{d}y_i\right)^{n-1}\mathrm{d}y_1\mathrm{d}y_2\cdots\mathrm{d}y_d$可写为

$$n\int_0^1\int_0^1\cdots\int_0^1\left(1-\prod_{i=0}^{d}y_i\right)^{n}-1\Big/-n\prod_{i=0}^{d-1}y_i\,\mathrm{d}y_1\mathrm{d}y_2\cdots\mathrm{d}y_{d-1}$$

$$=\int_0^1\int_0^1\cdots\int_0^1\sum_{k=1}^{n}(-1)^{k-1}\binom{n}{k}\left(\prod_{i=0}^{d-1}y_i\right)^{k-1}\mathrm{d}y_1\mathrm{d}y_2\cdots\mathrm{d}y_{d-1}$$

$$=\sum_{k=1}^{n}(-1)^{k-1}\binom{n}{k}\frac{1}{k^{d-1}}.$$

由双参数调和数的定义，$H_{d-1,n}=\sum_{k=1}^{n}(-1)^{k-1}\binom{n}{k}\frac{1}{k^{d-1}}$，因此，$\hat{C}_{d,n}=H_{d-1,n}$. 证毕.

至此，上述模型将文献[12]和文献[13]的工作的纳入到了一个统一的框架中，并证明了其在特定条件下的等价性；下一节，我们将设计一种新的数据划分方法，并采用上述概率模型来估计改进的数据划分方法所生成的 Skyline 基数. 值得注意的是，虽然下文的论述仍是基于稀疏性和独立性的假设，其目的在于简化论述的过程；论述的思路和结果并不

受稀疏性和独立性假设的限制，因而可推广到稠密数据和非独立分布的数据上(参见第 5 节).

3　基于排列的高效数据划分

本节提出了一种基于排列的数据划分方法；为证明其优越性，首先在二维空间中，证明了其局部 Skyline 基数的期望小于现有方法，接着再将上述结果推广到 d 维空间中.

3.1　划分算法

数据划分的主要任务是(1)将数据点映射到 $d!$ 个唯一的 d 维排列向量中，然后(2)将排列向量通过转换作为哈希值进行数据划分.

3.1.1　数据点映射

首先，对每一个数据点 $\bar{x}$ 进行空间映射，即为 $\bar{x}$ 的每一个维度 i 创建一个形如 $(F_{X_i}(x_i), i)$ 的键-值对，其中 $F_{X_i}(x_i)$ 为键，i 为值；这里暂不讨论如何获取概率分布函数 $F_X(\bar{x})$，相关内容会在 3.3 节中进行介绍. 据此，形成一个长度为 d 的键-值对数组 Q. 然后，根据键-值对的键 $F_{X_i}(x_i)$ 对该数组进行排序得到一个新的键-值对数组 S. 接着，从 S 中依次拷贝每一个键-值对的值，生成一个排列向量 $\boldsymbol{R}$，其第 i 个元素的值等于 S 的第 i 个键-值对的值.

表 2 展示了一个基于上述算法进行空间映射的处理过程的示例：假设当前已知有关数据点 $\bar{x}$ 的概率 $F_X\{\bar{x}\}$，构建的两个键-值对数组 Q 和 S 分别罗列在第 2 列和第 3 列，得到的排列向量 $\boldsymbol{R}$ 在第 4 列中.

表 2　数据点映射示例

$F_X(\bar{x})$	Q	S	$\boldsymbol{R}$
0.75	(0.75,1)	(0.25,2)	2
0.25	(0.25,2)	(0.50,3)	3
0.50	(0.50,3)	(0.75,1)	1

这种映射的几何意义在于，将数据空间分成了 $d!$ 个互不相交的子空间. 图 3 直观地展示了这种映射在 3 维空间中的几何释意，三个坐标轴分别对应了 3 维空间的三个维度，刻画了数据点 $\bar{x}$ 的概率 $F_X(\bar{x})$ 值，即 $F_{X_i}(x_i)$. 图中，整个数据空间被划分成了 6 个子空间，各由一个排列代表，如{1,2,3}和{3,1,2}等；如表 2 所示，假设数据点 $\bar{x}$ 在各维度上的概率值分别为 0.75、0.25 和 0.50，则 $\bar{x}$ 会被映射到对应于排列{2,3,1}的子空间中，也即构成了排列{2,3,1}对应的一个数据分块中的一个元素.

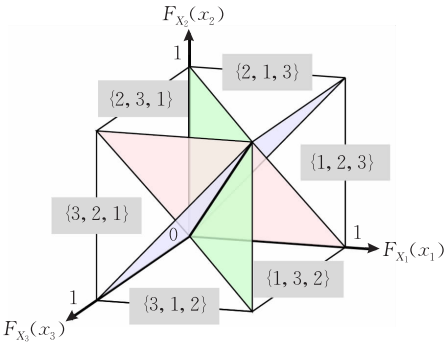


图 3　数据划分的几何释意

3.1.2　数据点划分

在完成空间映射之后，可直接采用排列向量作为哈希值并据此对数据点进行划分；即通过对前述排列向量 $\boldsymbol{R}$ 编码，生成一个正整数 ρ，表示数据点 $\bar{x}$ 所对应归属的分块. 在实现中，采用莱默编码①(Lehmer code)将一个包含 d 个元素的排列向量转成一个[1,$d!$]范围内的整数. 该操作等同于在排列向量基础之上进行二次映射.

具体地，对一个具有 d 个元素的排列，莱默编码用 0 到 $d-1$ 的一个数来描述它. 初始时，编码的第一个位置有 d 种选择，接着下一个位置除去初始元素之外有 $d-1$ 种选择，则该位置可以用一个 0 到 $d-2$ 之间的数来标识. 譬如，对于由元素{1,2,3,4,5}构成的排列〈3,1,5,2,4〉，第一个位置上的 3 表示 5 种选择({1,2,3,4,5})中的第 3 种，因此，为其分配索引 2；第二个位置上的 1 表示 4 种选择({1,2,4,5})中的第 1 种，因此，为其分配索引 0；第三个位置上的 5 表示 3 种选择({2,4,5})中的第 3 种，分配索引 2；第四个位置上的 2 表示 2 种选择({2,4})中的第 1 种，分配索引 0；第五个位置上的 4 只有一种选择，分配索引 0；最终，由此得到一个序列 $L=\{2,0,2,0,0\}$. 显然，这个序列的最后一位 L_d 总是为 0，倒数第二位 L_{d-1} 则是 0 或 1. 因此，莱默编码对应的整数可由 $\sum_{i=1}^{d-1} L_i(d-i)!$ 计算得到. 在上述例子中，排列〈3,1,5,2,4〉的莱默编码对应的整数为 $2\times24+0\times6+2\times2+0\times1=52$.

鉴于每个分块包含大量的数据点，可采用构建索引的方法来保存从排列到编码的映射，以保证通过快速查询的形式获取编码结果，最终形成一个模块化的方法 Encode(). 可见，在 Encode()方法中，从索引中查询返回映射结果的时间复杂度仅为 $O(d)$.

将空间映射和排列编码的思想结合起来，形

① https://en.wikipedia.org/wiki/Lehmer_code

成了针对单个数据点的划分算法，伪代码如算法 1 所描述. 按照如图 4 中的流程所示，第 1～5 行主要完成了对数据点的映射，而第 6～7 行则对数据点进行了编码，进而实现为该输入的数据点指定一个分块.

算法 1. PermPartition($\bar{x}$).

输入：$\bar{x}$ is a data point in d-dimensional space

输出：ρ is the partition that $\bar{x}$ belongs to

1. Q←an empty vector of key-value pairs;
2. for each dimensional value x_i of $\bar{x}$ do
3. insert $\{F_{X_i}(x_i), i\}$ into Q;
4. S←sort Q by the *key* value;
5. $\boldsymbol{R}$←a vector of length d such that $\boldsymbol{R}[i]=S[i]$.value;
6. ρ←Encode($\boldsymbol{R}$);
7. return ρ

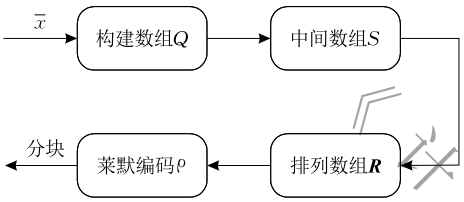


图 4 算法 1 的流程图

关于数据划分算法的性能，第 1～3 行的时间复杂度为 $O(d)$，其中 d 是数据的维度(常量)；第 4 和第 5 行的时间复杂度分别为 $O(\log d)$ 和 $O(d)$；第 6 行的时间复杂度为 $O(d)$；因此，算法 1 的整体时间复杂度为 $O(d)$.

3.2 理论分析

采用 2.2 节的概率模型，可从理论上估计上述数据划分方法的局部 Skyline 基数. 具体如下：首先，通过一个二维空间的例子阐述估计的过程，然后，将该过程扩展到 d 维空间.

考虑存在一个二维空间，假设其同时具备稀疏性和独立性，且式子 $F_X(\bar{x})=F_{X_1}(x_1)F_{X_2}(x_2)$ 和 $0\leq F_{X_1}(x_1), F_{X_2}(x_2)\leq 1$ 总是成立的. 则对于任意的 $\bar{x}=\{x_1,x_2\}\in\mathbb{R}^2$，其对应的概率向量 $\{F_{X_1}(x_1), F_{X_2}(x_2)\}$ 落在概率空间 $[0,1]^2$ 中. 由定理 3 可知，$\bar{x}$ 是一个 Skyline 点的概率等于

$$\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}(1-F_{X_1}(x_1)F_{X_2}(x_2))^{n-1}f_{X_1}(x_1)f_{X_2}(x_2)\mathrm{d}x_1\mathrm{d}x_2$$
$$=\int_0^1\int_0^1(1-F_{X_1}(x_1)F_{X_2}(x_2))^{n-1}\mathrm{d}F_{X_1}(x_1)\mathrm{d}F_{X_2}(x_2).$$

上式的几何意义如图 5 所示：若一个数据点是 Skyline 点，需要满足所有其他 $n-1$ 个点落在阴影区域；一个数据点落在阴影区域的概率等于该区域的大小.

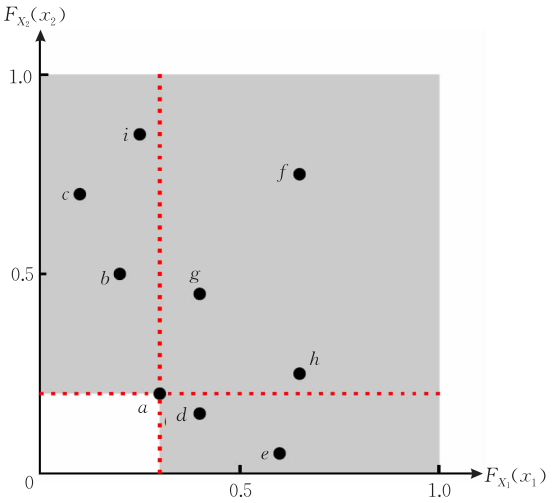


图 5 二维空间中的 Skyline 概率

文献[12]和文献[13]研究建议，在独立均匀分布下，采用 $x_1=x_2$ 作为二维空间划分的准则，但未能给出理论上的分析与证明，即采用如下的划分函数：

$$\rho(\bar{x})=\begin{cases}0, & x_1>x_2\\ 1, & x_1\leq x_2\end{cases}.$$

图 6 给出了采用 $x_1=x_2$ 进行划分的示例. 每一个数据点 $\bar{x}$ 依据 $\rho(\bar{x})$ 的取值被划分到分块 0 或者分块 1 中；其中，a、d 和 e 是分块 0 中的局部 Skyline 点，而分块 1 中的局部 Skyline 点为 c、b 和 g；经过最终的局部 Skyline 合并，得到全局 Skyline 为 a、b、c、d 和 e(如图 1 所示).

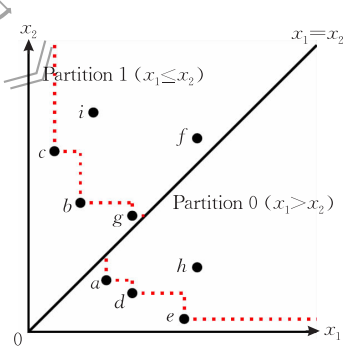


图 6 二维空间中现有划分示例

下面，对采用 3.1 节中的方法在二维空间中进行划分的情况进行分析，后续可以看到所提方法是将文献[12]和文献[13]中启发式的划分思想扩展到任意分布上，并给出局部 Skyline 基数期望的理论结果. 在任意分布的二维空间中，基于排列的数据划分方法采用通用的划分函数：

$$\rho(\bar{x})=\begin{cases}0, & F_{X_1}(x_1)>F_{X_2}(x_2)\\ 1, & F_{X_1}(x_1)\leq F_{X_2}(x_2)\end{cases}.$$

类似地，该函数将概率空间划分为两部分，如图 7 所示.

首先，考虑 $\rho(\bar{x})=0$ 所对应的分块 ρ_0 的局部

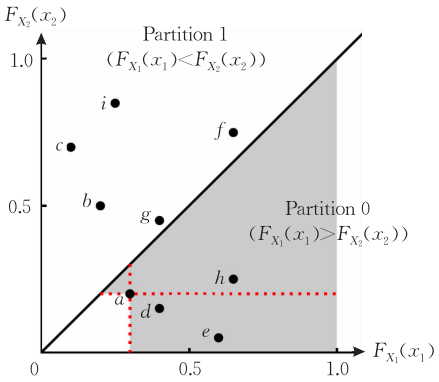


图 7　分块的 Skyline 概率

Skyline 基数期望. 由于当且仅当其他数据点落在阴影区域中时，一个数据点才是局部 Skyline 点. 因此假设 ρ_0 拥有 $n/2$ 个点，则 ρ_0 的局部 Skyline 基数的期望等于

$$E(SKY(\rho_0))=\frac{n}{2}\cdot\frac{\int_0^1\int_{y_2}^1\left(\frac{1/2-y_1y_2+y_2^2/2}{1/2}\right)^{n/2-1}\mathrm{d}y_1\mathrm{d}y_2}{1/2},$$

其中，$y_i\triangleq F_{X_i}(x_i)$. 由于二维空间中两个划分 ρ_0 和 ρ_1 是对称的，因此，$E(SKY(\rho_0))=E(SKY(\rho_1))$. 此时，两部分的局部 Skyline 基数的期望和等于

$$\begin{aligned}\hat{C}_{2,n,2}&=2n\int_0^1\int_{y_2}^1\left(\frac{1/2-y_1y_2+y_2^2/2}{1/2}\right)^{n/2-1}\mathrm{d}y_1\mathrm{d}y_2\\&=2\int_0^1\frac{(1-y_2^2)^{n/2}}{y_2}-\frac{(1-y_2)^n}{y_2}\mathrm{d}y_2\\&=2\left(\int_0^1\frac{1-(1-y_2)^n}{y_2}\mathrm{d}y_2-\int_0^1\frac{1-(1-y_2^2)^{n/2}}{y_2}\mathrm{d}y_2\right)\\&=2\sum_{i=1}^{n}\frac{1}{i}-\sum_{i=1}^{n/2}\frac{1}{i}=2\hat{C}_{2,n}-\hat{C}_{2,n/2}.\end{aligned}$$

将上述结果归纳总结，可得如下定理.

定理 4.　在稀疏性和独立性的假设下，若二维空间中采用划分函数

$$\rho(\bar{x})=\begin{cases}0, & F_{X_1}(x_1)>F_{X_2}(x_2)\\1, & F_{X_1}(x_1)\le F_{X_2}(x_2)\end{cases},$$

则局部 Skyline 基数的期望和等于

$$\hat{C}_{2,n,2}=2\hat{C}_{2,n}-\hat{C}_{2,n/2}.$$

定理 4 的推断是基于 $|\rho_0|=|\rho_1|=n/2$ 的假设，当点数 $n\gg 2$ 时，可有效估计局部 Skyline 基数. 相比之下，若采用随机划分，其局部 Skyline 基数期望和为 $\hat{C}^{\mathrm{rand}}_{2,n,2}=2\hat{C}_{2,n/2}$，由于 $2\hat{C}_{2,n}-\hat{C}_{2,n/2}<2\hat{C}_{2,n/2}$，因此，从期望上基于排列的方法可以生成比随机划分(或者基于网格①)更小的候选集.

上述针对二维空间的结果可以很容易地推广到 d 维空间中. 对于数据点 $\bar{x}=\{x_1,x_2,\cdots,x_d\}$，$d$ 维空间中总共有 $d!$ 个不同的排列方式，按照算法 1 可以快速地将数据点划分到各分块中.

定理 5.　在稀疏性和独立性假设下，d 维空间中采用基于排列的数据划分方法得到的局部 Skyline 基数的期望和满足：

$$\hat{C}^{\mathrm{perm}}_{d,n,d!}=d!n\int_0^1\int_{y_d}^1\cdots\int_{y_2}^1\left(\frac{1/d!-C_{\mathrm{perm}}}{1/d!}\right)^{n/d!-1}\mathrm{d}y_1\mathrm{d}y_2\cdots\mathrm{d}y_d,$$

其中，$C_{\mathrm{perm}}=\int_0^{y_d}\int_{t_d}^{y_{d-1}}\cdots\int_{t_2}^{y_1}1\mathrm{d}t_1\mathrm{d}t_2\cdots\mathrm{d}t_d$.

而随机划分的方法将 n 个数据点划分到 $d!$ 个分块中，每个分块拥有 $n/(d!)$ 个点，每个划分的局部 Skyline 基数是 $\hat{C}_{d,n/d!}$. 那么，随机划分中，局部 Skyline 基数的期望和为 $\hat{C}^{\mathrm{rand}}_{d,n,d!}=d!\hat{C}_{d,n/d!}$. 由于对于任意的 $y_i\in[0,1]$，有 $d!C^{\mathrm{perm}}>\prod_{i=1}^{d}y_i$，因此不等式 $\hat{C}^{\mathrm{perm}}_{d,n,d!}<\hat{C}^{\mathrm{rand}}_{d,n,d!}$ 显然是成立的. 这从理论上保证了基于排列的方法产生的数据划分是负载均衡的，且该方法的局部 Skyline 基数从期望上是小于随机划分(或者基于网格)方法.

为了对上述理论结果有一个直观的认识，图 8 展示了在不同的数量数据点上两种方法的局部 Skyline 基数的期望和，同时还给出了全局 Skyline 点数作为参考. 可以看到，基于排列的方法生成的局部 Skyline 基数的期望和随着全局 Skyline 基数的增长呈亚线性(sublinearly)增长，而随机划分方法的结果则是超线性(superlinearly)增长；换言之，基于排列的方法有效减少了局部 Skyline 所生成的假阳性候选数据点的可能性.

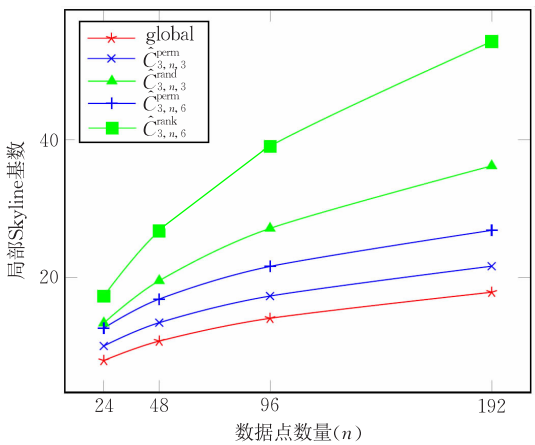


图 8　局部 Skyline 基数的期望和

① 从期望上，基于网格的方法具有和随机划分方法相同的局部 Skyline 基数.

3.3 分布函数拟合

接下来讨论如何估计概率分布 $F_X(\bar{x})$ 以将 x_i 转换成 $F_{X_i}(x_i)$. 在各维度之间相互独立且服从相同的分布的情况下,由于 $F_{X_i}(x_i)$ 的单调性,$F_{X_i}(x_i)\leq F_{X_j}(x_j)\Leftrightarrow x_i\leq x_j$,可以直接使用 x_i 的值并利用上述模型进行基数的估计. 其他情况下,则需要进行数据预处理来转换得到 $F_{X_i}(x_i)$. 一种相对直接的方法是将所有数据点按照每个维度的值进行排序,然后取数据点 $\bar{x}$ 在该序列中的排位与数据点总量的比值(归一化)作为概率 $F_{X_i}(x_i)$,即

$$F_{X_i}(x_i)=\frac{|\{\bar{y}\,|\,y_i<x_i\}|}{n},\ \bar{x},\bar{y}\in G.$$

这种方法步骤简单,但是面对大规模数据集时的开销较大,时间复杂度为 $O(dn\log n)$. 鉴于此,受直方图技术启发,针对大规模数据考虑一种结合抽样统计信息的估计方法来测算数据点的排位.

具体地,针对某一维度,首先通过简单随机抽样(每个数据点都具有相同的抽中概率)创建一组连续的数值区间,称之为桶(记作 B_s);然后将数据点的维度值放入桶中,并计算每一个桶中的数据点个数 $|B_s|$. 给定一个在区间 $[\min(B_s),\max(B_s)]$ 里的维度 i 上的值 x_i,其中 $\min(B_s)$ 和 $\max(B_s)$ 分别是 B_s 上的最小值和最大值. 那么,x_i 的归一化之后的排位可估计为

$$F_{X_i}(x_i)=\frac{x_i-\min(B_s)}{\max(B_s)-\min(B_s)}\cdot\frac{|B_s|}{n}+\frac{\sum_{k=1}^{s-1}|B_k|}{n}.$$

最后,所有落在各个区间 $[\min(B_s),\max(B_s)]$ 中的数据点都被映射到区间

$$\left[\frac{1}{n}\sum_{k=1}^{s-1}|B_k|,\frac{1}{n}\sum_{k=1}^{s}|B_k|\right]$$

上,进而估计得出数据点对应的概率.

在并行环境中,$[\min(B_s),\max(B_s)]$ 和 $|B_s|$ 等有关桶 B_s 的统计信息会通过广播的形式发送给所有工作者,然后各工作者便可并行地将本地拥有的数据点进行预处理.

3.4 局部和全局 Skyline 计算

SFS 算法不仅可用在局部 Skyline 计算中,在合并局部 Skyline 的阶段,同样可采用 SFS 算法进行计算来得到全局 Skyline. 因此,在局部和全局 Skyline 计算中,均采用经典的 SFS 算法(参见 2.1 节). 值得注意的是,设计新的集中式 Skyline 处理算法不属于本研究范畴.

总结而言,本节所提出的基于排列的数据划分算法主要是针对并行 Skyline 计算的过滤阶段进行了合理的过程重构:

(1) 预处理. 对于数据空间中的每一个维度 i,利用抽样统计信息,将 $\bar{x}$ 的取值 x_i 预处理成一个在 $[0,1]$ 区间上的概率 $F_{X_i}(x_i)$;换言之,对于每一个 $\bar{x}$,生成一个相应的概率向量表示 $F_X(\bar{x})$;

(2) 数据划分. 将每个预处理后的概率向量映射到一个排列上,再通过对排列向量编码转换,将 $\bar{x}$ 分到某个分块中;

(3) 局部计算. 各工作者利用经典的 SFS 算法分别计算其拥有的局部数据的 Skyline.

最后,过滤阶段的局部候选点集交由合并阶段处理,并再次利用 SFS 算法计算得到全局 Skyline.

4 实验与分析

本节主要通过大规模的实验来评估所提方法的有效性和先进性.

4.1 实验设置

实验在一个具有 16 个节点的计算集群上开展的,每个节点具备 2 个 Intel Xeon E5530 2.5GHz 和 24GB 内存,运行环境为 Debian GNU/Linux 5.0.

算法. 实验中,主要对比了两个现有的并行 Skyline 算法[10,14],分别称作 Angle[10] 和 Proj[14]. 实验中并没与 Park 等人提出的算法[5] 对比,这是因为他们的方法仅使用了一个 Reduce 任务进行全局 Skyline 合并;而在本文的设定中,一个计算节点的内存是无法容纳所有的候选 Skyline 点. 实验也没有对比 Tao 等人提出的算法[15],因为他们的方法不支持处理二维以上数据. 对于所有的算法,每个计算节点上的局部 Skyline 及全局 Skyline 合并计算均采用了经典的 SFS 算法[13].

数据集. 实验中,同时采用了人工数据和真实数据. 人工数据来自服从独立分布和负相关分布的数据生成器,生成的数据集大小从一百万(10^6)到十亿(10^9),数据点的维度数在 $[2,6]$ 之间;实验默认的数据集大小为一百万. 真实数据则采用了著名的 NBA① 和 Household② 数据集.

实现细节. 实验中,采用了脱机(Standalone)的实现方式,原因主要在于现有算法 Angle 和 Proj 在

① http://www.nba.com
② http://www.ipums.org

数据分块的大小上没有保证，容易导致负载不均衡并造成系统崩溃，而脱机实现可以减轻上述问题. 所有相关算法都使用 C++语言实现，g++4.3.2 编译.

脱机实验是按如下规程设计的：(1) 一个集中式的算法依照不同的数据划分策略(基于角度的 Angle、基于超平面的 Proj 和基于排列的 Perm)将数据进行分块；该步耗时记作划分时间；(2) 利用 SFS 算法顺序处理每一个划分，单个分块所需的最长处理时间作为该步的耗时；(3) 利用 SFS 算法合并处理上一步的输出结果，该步的耗时也记录下来. 上述三个步骤的耗时之和为总体响应时间. 下文的实验结果分析中，将每组数据集上的结果分成“数据划分效果”和“时间性能”两部分论述，前者考查了数据划分的对负载均衡和局部 Skyline 基数的影响，后者对比分析了各算法的数据划分耗时和整体响应时间.

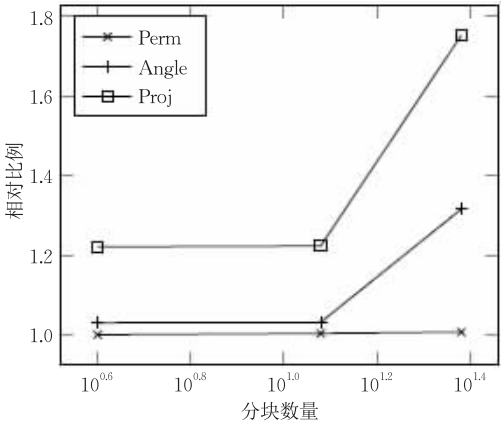
4.2　实验结果及分析

4.2.1　独立分布人工数据集

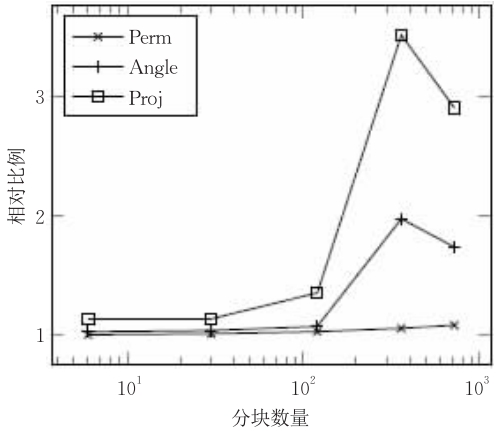
(1) 数据划分效果

图 9(a)和(b)展示了在不同的分块数量条件下，划分得到的最大分块的大小. 最大分块的大小意味着一个计算节点的内存大小，只有满足该大小才能满足存储局部 Skyline 点的需求. 当划分在理想状态下，即负载均衡，每个分块的大小为 n/m，其中 n 是数据点个数，m 是分块的数量. 因此，n/m 可以作为评价划分相对效果的一个基准. 将该大小作为 100%，对比发现，Perm 的划分效果相比其他方法更加接近 100%. 而 Angel 和 Proj 则需要 190% 和 350%的理想内存大小. 注意到图 9(b)中，Angle 和 Proj 在分块数位于 10^2 和 10^3 的区间上出现了一个拐点，其主要原因可能是这两个算法的运行时间主要取决于处理速度最慢的一个分块，而当分块数量增加后，由于它们的负载均衡能力和伸缩性较差，实验结果也变得不可预测.

图 10(a)和(b)展示了所有分块上的最大局部 Skyline 基数. 局部 Skyline 基数和局部 Skyline 计算的开销是成正比的；换言之，局部 Skyline 基数越大，则局部 Skyline 计算的开销越大. 该实验中，Perm 得到的局部 Skyline 基数在所有参数设定下均远小于其他方法.

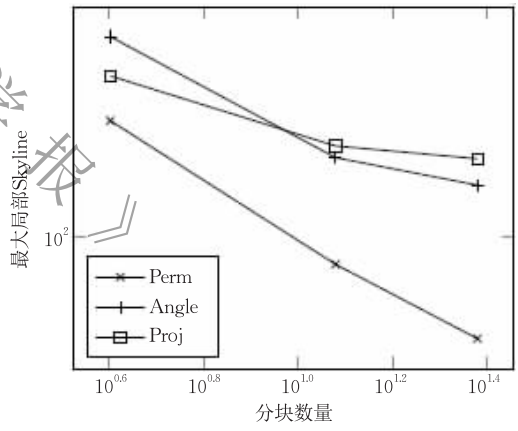


(a) dim=4

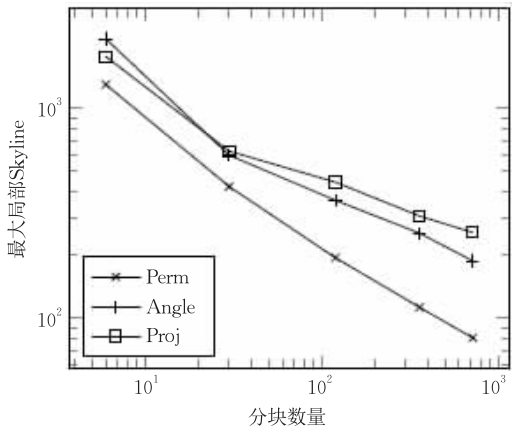


(b) dim=6

图 9　独立分布数据集上数据划分实验结果 I



(a) dim=4



(b) dim=6

图 10　独立分布数据集上数据划分实验结果 II

(2)时间性能

首先,图 11(a)和(b)统计了数据划分的耗时情况. Perm 具有和分块数量相同的划分时间,而 Angle 和 Proj 则分别慢于 Perm 10 倍和 5 倍. Angle 和 Proj 的额外开销,这主要是为了追求均衡负载而做的处理;相比而言,Perm 则是天生负载均衡的,因而无须付出额外代价. 对于 Proj 而言,Angle 更加耗时,因为它需要将笛卡尔坐标空间转换到超球面空间. Perm 的天生负载均衡且划分高效的特质意味着 Perm 算法能够横向扩展到更多的计算节点.

其次,图 11(c)和(d)则展示了各算法消耗在局部 Skyline 计算上的时间,Perm 在较大的分块数量上的性能优势较为明显,也就是说 Perm 可以有效提高并行化程度.

最后,图 11(e)和(f)描绘了各算法的总体响应时间,该时间和分块大小及局部 Skyline 基数都有关联. Perm 在划分大小和局部 Skyline 基数上都是均衡的,且在前述实验中都优于 Angle 和 Proj,因

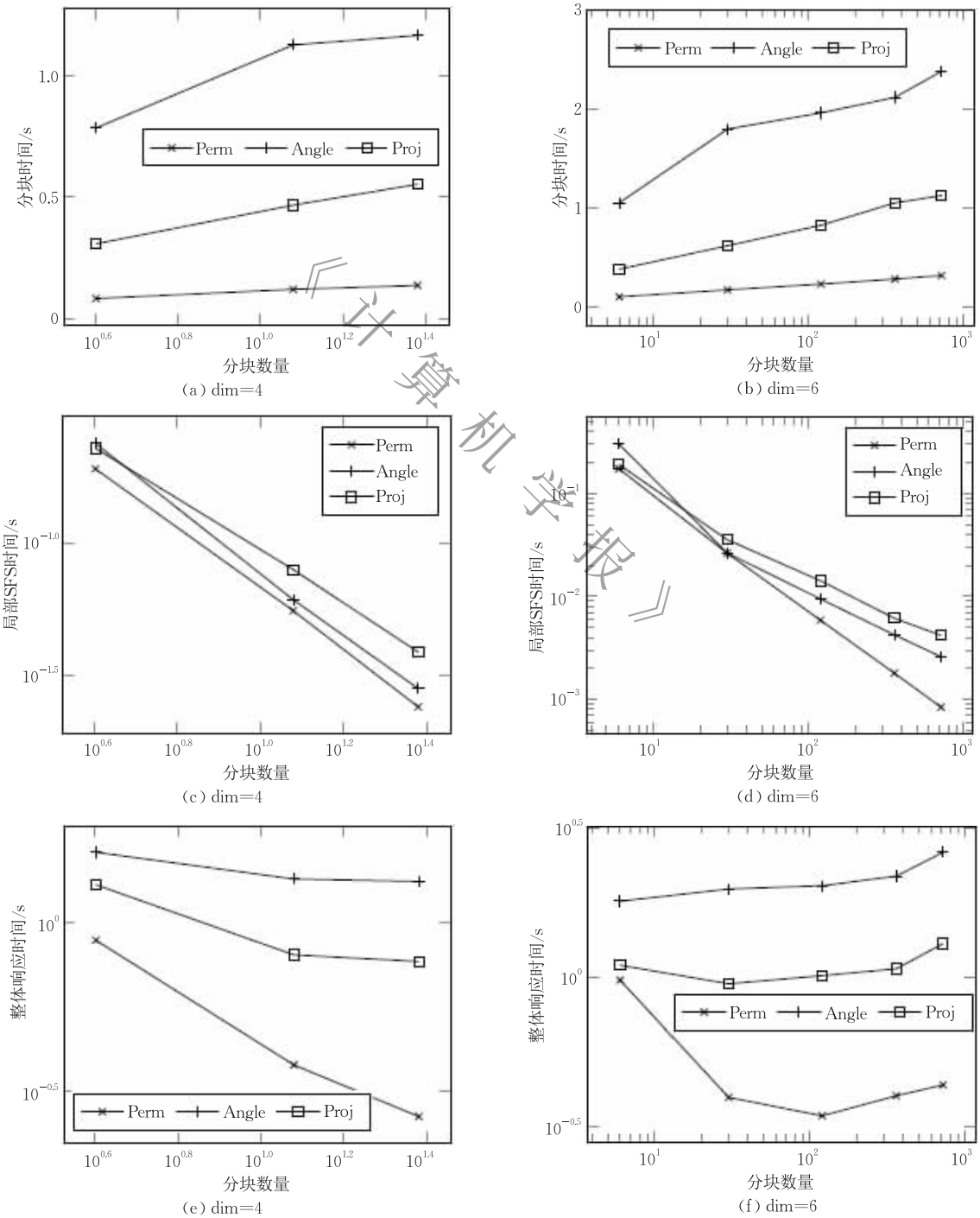


图 11 独立分布数据集上时间性能实验结果

此，它在所有参数设定下的性能均优于其他算法，该结果也是符合预期的. 此外，虽然 Angle 在数据划分阶段比 Proj 耗时，但是在总体时间上，Angle 在绝大多数参数设定下优于 Proj，原因是 Angle 在划分大小和局部 Skyline 基数都小于 Proj. 这亦说明数据划分在整个并行 Skyline 计算中的重要性，同时也从另一个侧面阐明了 Perm 优于其他同类算法的本质原因所在. 另外，注意到不同的维度上，算法表现出了不同的趋势；尤其是在高维数据上，所有算法的整体响应时间会随着分块数的增加而有所增加，其原因可能是高维空间给数据处理带来了一些困难，特别是在汇总分块结果的全局 Skyline 阶段需要花费更多的时间. 这也说明并行 Skyline 计算的并行数量是存在一个合理的上限的，超过该上限之后，并行化所带来的优势就可能会被汇总的代价所抵消掉了.

4.2.2　非独立分布人工数据集

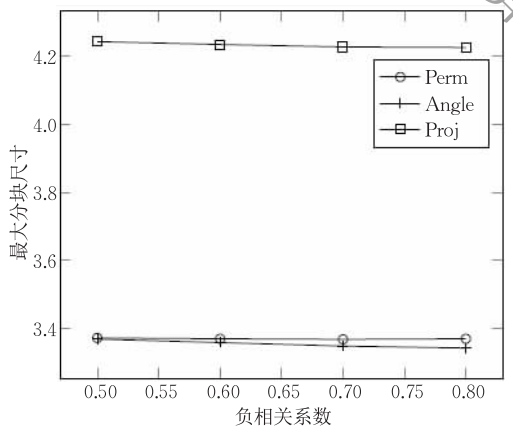
除了独立分布之外，实验还考查了在负相关分布下的表现，以维度数为 6 的数据为例. 负相关分布定义了数据点靠近超平面 $\sum_{i=1}^{d} x_i = d/2$ 的程度，其可能的最大距离是 $\sqrt{d}/2$；负相关系数为一个 $[0,1]$ 之间的实数，使用人工数据生成器构建了 7 个负相关系数在 0.5 到 0.8 范围之内的人工数据集.

(1) 数据划分效果

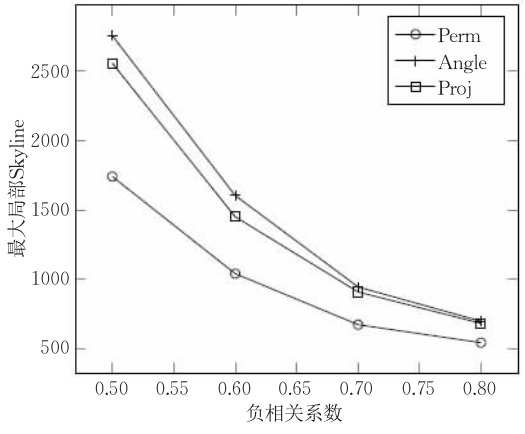
图 12(a)和(b)分别展示了最大的分块所包含的数据点的个数以及最大的局部 Skyline 基数. Proj 在数据划分上的效果相对较差，Perm 和 Angle 则比较接近；然而，在最大局部 Skyline 基数上，所有算法的结果均随着负相关系数的增大而减小，且 Perm 显示出较明显的优势.

(2) 时间性能

图 13(a)和(b)分别展示了各算法在数据划分和总体时间消耗上的情况. 在数据划分阶段，各算法的性能都比较稳定，从快到慢的依次排序为 Perm、Proj 和 Angle；从总体时间消耗看，各算法的效率随着负相关度的增加而提升，从快到慢的依次排序为 Perm、Proj 和 Angle.

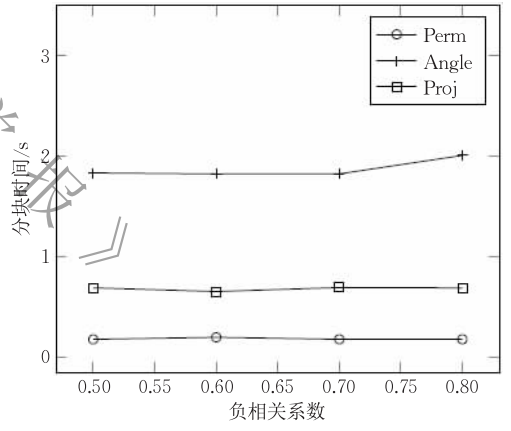


(a) dim=6

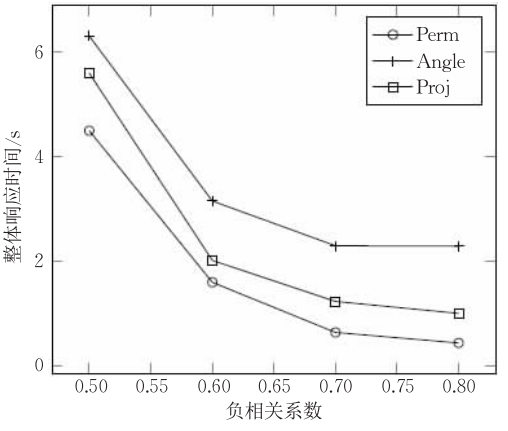


(b) dim=6

图 12　非独立分布数据集上数据划分实验结果



(a) dim=6



(b) dim=6

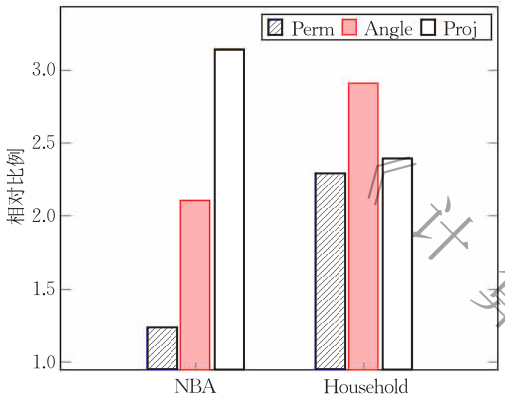
图 13　非独立分布数据集上时间性能实验结果

4.2.3 真实数据集

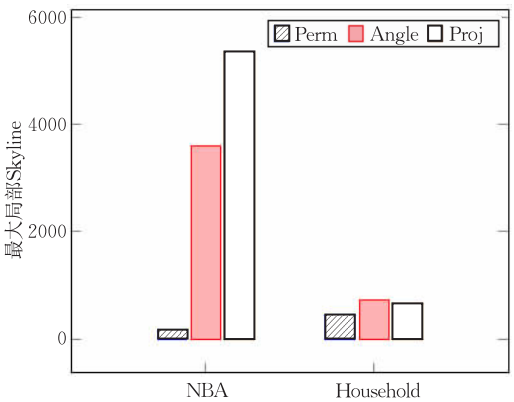
在 NBA 和 Household 两个数据集上，Perm 都展示出相比 Angle 和 Proj 更加稳定和优越的性能.

(1)数据划分效果

图 14(a)展示了两个数据集上各算法生成的最大分块的大小. 从图中可见，无论是 NBA 还是 Household 数据集上，Perm 都是负载均衡最好的方法，相比 Angle 和 Proj 有很显著地改善. 接着，图 14(b)给出了各种算法生成的最大局部 Skyline 基数的大小. 在这项结果上，Perm 依旧优势明显；并且由于 NBA 数据集具备更强的负相关性，因此，Perm 在 NBA 上的优越性相比在 Household 上更加显著.



(a) 负载均衡

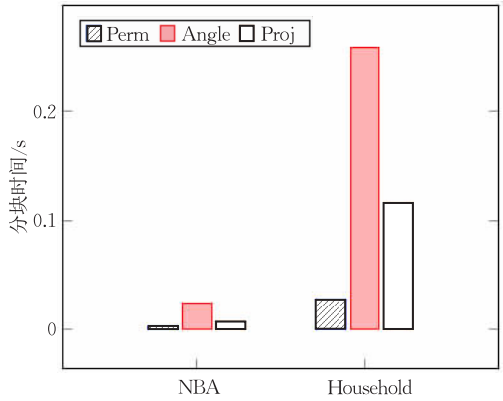


(b) 最大局部Skyline

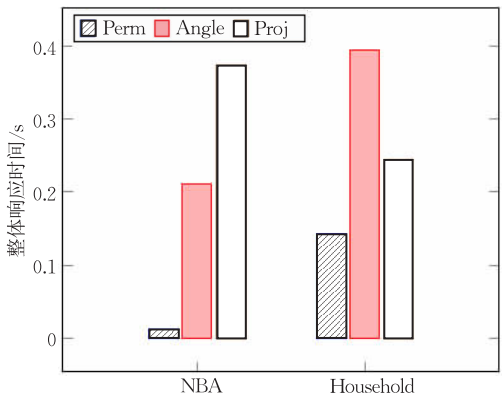
图 14 真实数据集数据划分实验结果

(2)时间性能

然后，图 15(a)绘制了各算法在划分上消耗的时间. 因为 Perm 采用了高效的划分策略(配合映射索引)，其花费在数据划分上的时间极少；相较而言，Angle 的划分时间开销较大，原因在于其复杂的坐标转换方法. 最后，图 15(b)描绘了各算法在整个 Skyline 计算过程中的消耗时间. 基于前序实验结果，该组实验结果也是可期的，看到 Perm 在两个数据集上都表现出很大的性能优势，存在至少一个数量级的执行时间提升.



(a) 数据划分时间



(b) 整体响应时间

图 15 真实数据集时间性能实验结果

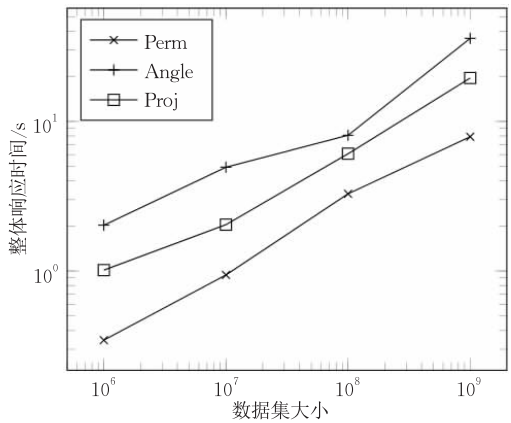
总结而言，无论是 NBA 还是 Household 数据集，Perm 都是最优的选择. 特别是在 NBA 数据集上，Perm 相比 Angle 和 Proj 有至少一个数量级的优势，因为 NBA 中的数据点之间存在负相关性，一个数据划分方法如果能较好地对数据进行分块，则能生成较小的局部 Skyline，进而直接影响并行算法的整体效率. Angle 和 Proj 由于不能很好地进行数据划分，导致若干数据分块上的局部 Skyline 基数较大，这些分块上的 SFS 算法的响应时间较长，最终导致整体性能欠佳. 在 Household 数据集上，Perm 比 Angle 和 Proj 分别快 1.9 倍和 1.2 倍. 由于 Household 数据集是一个相关性较高的数据集，所有算法在局部 Skyline 计算中的性能都是令人满意的；而 Perm 在这个阶段的额外优势在于高效的数据划分上，换言之，良好的负载均衡和最小化局部 Skyline 基数保证了并行 Skyline 计算的效能.

4.2.4 可扩展性实验

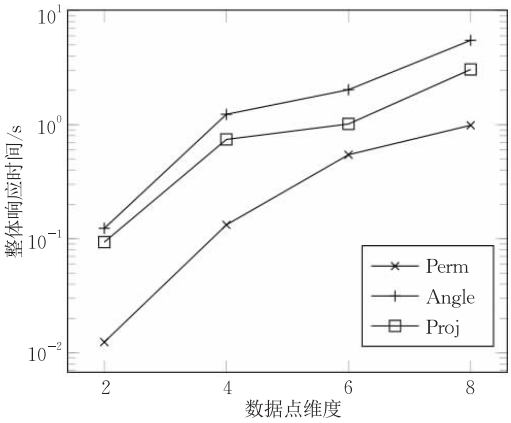
在独立分布的人工数据集上开展了可扩展性的实验，评测所提算法针对数据集大小变化和数据点的维度数变化时所表现出来的性能趋势.

首先，评测所提算法针对数据集的可扩展性，固定算法进行数据分块数量为 120. 从图 16(a)中可见，随着数据集规模的增长(从 10^6 到 10^9)，Perm 算法的时间响应性能匀速下降，未遇见明显的拐点等异常；相较而言，Angle 算法在若干数据集上存在异常拐点，时间性能可能不遵循一般规律和预期. 其主要原因在于，Perm 算法的效果有理论保证的，良好的数据划分保证了各分块的负载均衡，因此，能够保证算法对数据集的规模具有良好的可扩展性.

然后，考查所提算法针对数据点维度数的可扩展性，此实验中采用默认规模的数据集并固定算法进行数据分块数量为 1209，数据点的维度在 2 到 8 之间变化. 从图 16(b)中可见，随着数据点维度的增加，算法处理的难度也逐渐增大，所有算法的响应时间均不同程度地增长. 注意到，Perm 算法的增长曲线是最平缓的，即具有最小的增长率. 该现象符合实验的预期，其主要原因在于，Perm 良好的数据划分效果保证了各分块的负载均衡，尽可能地规避了个别分块过长的处理响应时间，从而使得，在常见的数据维度空间中①，Perm 相对 Angle 和 Proj 在关于数据点维度方面具有更佳的可扩展性.



(a) dim=6, 1M～1000M



(b) dim=2～8, 1M

图 16　可扩展性实验结果 IV

5　扩展与讨论

第 2 节中的理论分析假设了数据的独立性，同时还要求了稀疏性. 其实，可将模型扩展到处理稠密数据的情况，即每个维度上允许存在重复值. 在这类应用中，通常维度的值域是十分有限的，譬如，男士泳衣的样式可能的取值包括“三角短裤”、“及膝短裤”、“过膝长裤”和“全身泳服”. 当维度数值被映射到一个可数集合上时，一个可数集合的分布是离散分布.

离散分布下，为 $\bar{x}\in\mathbb{R}^d$ 定义联合概率分布函数

$$p_X(\bar{x})=P(X=\bar{x}),$$

其中，时间 $X=\bar{x}$ 是如下定义的：

$$\{X=\bar{x}\}=\{X_1=x_1,X_2=x_2,\cdots,X_d=x_d\}$$
$$=\bigcap_{k=1}^{d}\{X_k=x_k\}.$$

那么，$F_X(\bar{x})=\sum_{\bar{y}\le\bar{x}}(\bar{y})$，即

$$F_{X_1,X_2,\cdots,X_d}(x_1,x_2,\cdots,x_d)=$$
$$\sum_{y_1\le x_1}\sum_{y_2\le x_2}\cdots\sum_{y_d\le x_d}p_{X_1,X_2,\cdots,X_d}(y_1,y_2,\cdots,y_d),$$

其中，$x_1,x_2,\cdots,x_d\in\mathbb{R}$，$y_1,y_2,\cdots,y_d\in\mathbb{R}$.

对于任意数据点 $\bar{x}$，其他数据点支配 $\bar{x}$ 的概率等于 $F_X(\bar{x})-p_X(\bar{x})$. 基于类似定理 3 的思路，可推得如下结果.

定理 6.　在稠密性假设下，Skyline 基数的期望等于

$$\hat{C}_{d,n}=n\sum_{\bar{x}\in\mathbb{R}^d}p_X(\bar{x})[1-(F_X(\bar{x})-p_X(\bar{x}))]^{n-1}.$$

对于离散分布而言，概率函数 $p_X(\bar{x})$ 是多数情况下都是非零的，且有 $0\le p_X(\bar{x})\le F_X(\bar{x})\le 1$. 当 n 趋向于无穷大时，可进一步推得如下结论.

定理 7.　在稠密性的假设下，Skyline 基数的期望与数据点个数 n 比值的极限等于

$$\lim_{n\to+\infty}\frac{\hat{C}_{d,n}}{n}=\sum_{\bar{x}\in\mathbb{R}^d\wedge F_X(\bar{x})=p_X(\bar{x})}p_X(\bar{x}).$$

注意到，文献[11]提出，在独立性假设下，当各维度服从一个 p 值的均匀分布时，那么，

① 主流的 Skyline 处理主要针对 2～6 维的数据点进行，并已能够满足多数现实应用的需求；有关更高维度数据的处理，或需要结合相应的数据约减和降维技术开展研究，超出了本研究的范畴.

$$\lim_{n\to+\infty}\frac{\hat{C}_{d,n}}{n}=\frac{1}{p^d}.$$

定理7的结论将上述结果又推进了一步，可适用于任意分布，且免除了有关独立性的约束.

6 相关工作

Skyline是数据分析和决策支持应用中一个十分重要的基础问题，近年来得到了大量研究的关注，譬如，结合预处理或不带预处理的面向Skyline算子的索引结构，外存计算，ad-hoc计算，连接计算等，以及用Skyline算计处理偏斜的和不确定的数据等. 文中关注的是单机无法处理的针对大数据集的并行和分布式处理.

6.1 并行Skyline计算

针对并行Skyline计算，首先采用的是随机数据划分方法[16]，而后是基于网格的数据划分方法[9]（也应用在分布式Skyline计算[8]）. 相对于无共享(share-nothing)的架构，Park等人的研究聚焦在多核(multi-core)架构上[17]，其他类似的工作还包括[18-19]. 在大数据处理结构方面，Zhang等人的研究首次实现了一个基于MapReduce编程框架的Skyline算子[6]，并设计了三个算法——MR-BNL、MR-SFS和MR-Bitmap. 同期，丁琳琳等人研究提出了四个MapReduce框架下的Skyline计算方法[20]，对候选数据点进行多种形式的过滤. 之后，Park等人的研究提出用quad-tree在局部Skyline计算步骤中对非Skyline数据点进行过滤，在数据划分上，该方法采用了基于网格的划分[5]. Tao等人的研究考查了利用MapReduce进行排序和滑动聚合的问题[15]；作为一个副产品，还给出了一种针对二维数据的高效Skyline算法，但无法扩展到更高的维度空间.

新近的研究在约减局部Skyline基数方面有较大进展. Vlachou等人的研究提出一种基于角度的划分方法[10]，主要利用了超球面(hyper-spherical)坐标；最近，基于角度的划分方法被扩展实现到基于MapReduce的并行框架上[21]，但其在数据划分方面的缺陷并没有得到重视和解决. Köhler等人注意到这种基于角度的超球面坐标变换可能相当耗时，因此提出了一种向超平面投影的划分方法[4]. 上述两种算法均可以纳入到文中的通用并行Skyline计算框架中，但是它们在负载均衡方面的缺陷限制了系统的性能和可扩展性. 针对偏序关系的数据应用，Yin和Gu研究了如何将偏序关系映射到全序域中再进行并行Skyline计算的问题[22]，其方法可通过改造运用到本文所提的算法上，使其具备处理偏序数据的能力；针对高维空间，周红福等人研究了在线的集中式的高效子空间Skyline算法CSky[23]，其渐进性的算法优势主要得益于一种新颖的数据结构InvertS.

6.2 分布式Skyline计算

分布式Skyline计算同样十分关注数据的划分问题，但其设计的出发点与网络结构存在较大关联. Balke等人[24]和Lo等人[25]的研究采用了纵向划分方法，他们的方法与采用横向划分的并行Skyline算法存在较大差别.

此外，一个相关方向的研究是针对ad-hoc网络拓扑结构（基于树[26]和全连接[27]）的分布式环境中如何高效执行Skyline计算. 感兴趣的读者可以延伸阅读Hose和Vlachou的优秀综述[28].

6.3 集中式Skyline计算

相比经典的基于Block-Nested-Loop(BNL)的算法，SFS算法具有易优化、可管道化、输出有序等优点，且对计算节点内存的要求相对较小. 因此，SFS算法在大多数情况下通常能够提供相对BNL算法更佳的时间响应和系统性能.

7 结 论

文中主要研究了面向大数据的并行Skyline计算问题. 现有并行Skyline计算方法仍存在无法实现负载均衡或局部Skyline基数较大的缺陷，无法满足当前大数据处理与分析应用中在系统响应方面的要求. 首先，本文从理论上深入探究了一种用于估计Skyline基数的概率模型. 以此为基础，为了设计面向大数据集高效并行Skyline算子，提出了一种基于排列的数据划分方法. 这种方法不仅能够实现负载均衡，同时每个划分所生成的局部Skyline都具有较小的基数，从而提升系统的整体性能. 从理论分析和实验验证两个角度同时展示了所提方法的高效性；相比现有方法，新的并行Skyline算子在绝大多数参数设定下面具有更加优越的时间性能.

下一步，计划将所提技术扩展到其他相似的并行查询与分析处理中；譬如，考虑所提技术是否能够运用到并行大图计算上，以提升大图数据处理与分析的效率. 大图数据处理是当前数据管理领域的一个热点，已有一系列工作研究和讨论并行的图数据处理

问题，如并行的大图搜索与评估[29]等. 譬如，以图结构相似度查询[30]来举例，一个大图数据被划分成不同的分块是进行并行图处理的前提条件，而如何进行合理高效的分块则很大程度上决定了查询处理的性能. 为此，可考虑能否将计算量与候选集规模尽可能均衡的思想贯彻到并行大图处理的数据划分过程中，从而提高整体数据查询与分析的效果和效率.

致　谢　感谢匿名评审和编辑对本文质量的改进提供了大量建设性的意见和帮助. 同时感谢 SFS 算法作者对源代码的开源与共享！

参　考　文　献

[1] Meneghetti N, Mindolin D, Ciaccia P, Chomicki J. Output-sensitive evaluation of prioritized Skyline queries//Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data. Melbourne, Australia, 2015: 1955-1967

[2] Nagendra M, Candan K S. Efficient processing of Skyline-join queries over multiple data sources. ACM Transactions on Database Systems, 2015, 40(2): 10

[3] Shang H, Kitsuregawa M. Skyline operator on anti-correlated distributions. Proceedings of the VLDB Endowment, 2013, 6(9): 649-660

[4] Köhler H, Yang J, Zhou X. Efficient parallel Skyline processing using hyperplane projections//Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data. Athens, Greece, 2011: 85-96

[5] Park Y, Min J, Shim K. Parallel Computation of Skyline and reverse Skyline queries using MapReduce. Proceedings of the VLDB Endowment, 2013, 6(14): 2002-2013

[6] Zhang B, Zhou S, Guan J. Adapting Skyline computation to the MapReduce framework: Algorithms and experiments//Proceedings of the 2011 DASFAA International Workshops. Hong Kong, China, 2011: 403-414

[7] Wu P, Zhang C, Feng Y, et al. Parallelizing Skyline queries for scalable distribution//Proceedings of the 2006 International Conference on Extending Database Technology. Munich, Germany, 2006: 112-130

[8] Wang S, Ooi B C, Tung A K H, Xu L. Efficient Skyline query processing on peer-to-peer networks//Proceedings of the 2007 IEEE International Conference on Data Engineering. Istanbul, Turkey, 2007: 1126-1135

[9] Afrati F N, Koutris P, Suciu D, Ullman J D. Parallel Skyline queries//Proceedings of the 2012 International Conference on Database Theory. Berlin, Germany, 2012: 274-284

[10] Vlachou A, Doulkeridis C, Kotidis Y. Angle-based space partitioning for efficient parallel Skyline computation//Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data. Vancouver, Canada, 2008: 227-238

[11] Godfrey P. Skyline cardinality for relational processing//Proceedings of the 2004 International Symposium on Foundations of Information and Knowledge Systems. Wilhelminenburg Castle, Austria, 2004: 78-97

[12] Chaudhuri S, Dalvi N N, Kaushik R. Robust cardinality and cost estimation for Skyline operator//Proceedings of the 2006 IEEE International Conference on Data Engineering. Atlanta, USA, 2006: 64

[13] Chomicki J, Godfrey P, Gryz J, Liang D. Skyline with presorting: Theory and optimizations//Proceedings of the 2005 International Conference on Intelligent Information Systems. Gdansk, Poland, 2005: 595-604

[14] Köhler H, Yang J, Zhou X. Efficient parallel Skyline processing using hyperplane projections//Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data. Athens, Greece, 2011: 85-96

[15] Tao Y, Lin W, Xiao X. Minimal MapReduce algorithms//Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data. New York, USA, 2013: 529-540

[16] Cosgaya-Lozano A, Rau-Chaplin A, Zeh N. Parallel computation of Skyline queries//Proceedings of the 2007 Annual International Symposium on High Performance Computing Systems and Applications. Saskatoon, Canada, 2007: 12

[17] Park S, Kim T, Park J, et al. Parallel Skyline computation on multicore architectures//Proceedings of the 2009 IEEE International Conference on Data Engineering. Shanghai, China, 2009: 760-771

[18] Bøgh K S, Chester S, Assent I. Work-efficient parallel Skyline computation for the GPU. Proceedings of the VLDB Endowment, 2015, 8 (9): 962-973

[19] Chester S, Sidlauskas D, Assent I, Bøgh K S. Scalable parallelization of Skyline computation for multi-core processors//Proceedings of the 2015 IEEE International Conference on Data Engineering. Seoul, South Korea, 2015: 1083-1094

[20] Ding Lin-Lin, Xin Jun-Chang, Wang Guo-Ren, Huang Shan. Efficient Skyline query processing of massive data based on Map-Reduce. Chinese Journal of Computers, 2011, 34(10): 1785-1796 (in Chinese)
(丁琳琳，信俊昌，王国仁，黄山. 基于 Map-Reduce 的海量数据高效 Skyline 查询处理. 计算机学报，2011，34(10)：1785-1796)

[21] Miyoung J, Youngho S, Jae-Woo C. A parallel computation of Skyline using multiple regression analysis-based filtering on MapReduce. Distributed and Parallel Databases, 2017, 35(3-4): 383-409

[22] Yin B, Gu K. Parallel Skyline computation for partially ordered domains//Proceedings of the 2017 IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications. Guangzhou, China, 2017: 699-706

[23] Zhou Hong-Fu, Gong Xue-Qing, Zheng Kai, Zhou Ao-Ying. CSky: An online efficient algorithm for subspace skyline computation in high dimensional space. Chinese Journal of Computers, 2007, 30(8): 1409-1417(in Chinese)
(周红福，宫学庆，郑凯，周傲英. 基于高维空间的在线高效子空间 Skyline 算法——CSky. 计算机学报，2007，30(8)：1409-1417)

[24] Balke W, Güntzer U, Zheng J. X. Efficient distributed Skylining for Web information systems//Proceedings of the 2004 International Conference on Extending Database Technology. Heraklion, Greece, 2004: 256-273

[25] Lo E, Yip K Y, Lin K, Cheung D W. Progressive Skylining over Web-accessible databases. Data and Knowledge Engineering, 2006, 57(2): 122-147

[26] Cui B, Chen L, Xu L, et al. Efficient Skyline computation in structured peer-to-peer systems. IEEE Transactions on Knowledge and Data Engineering, 2009, 21(7): 1059-1072

[27] Rocha-Junior J B, Vlachou A, Doulkeridis C, Nørvåg K. Efficient execution plans for distributed Skyline query processing//Proceedings of the 2011 International Conference on Extending Database Technology. Uppsala, Sweden, 2011: 271-282

[28] Hose K, Vlachou A. A survey of Skyline processing in highly distributed environments. The VLDB Journal, 2012, 21(3): 359-384

[29] Fan W, Hu C. Big graph analyses: From queries to dependencies and association rules. Data Science & Engineering, 2017, 2(1): 36-55

[30] Zhao X, Xiao C, Lin X, et al. Efficient structure similarity search: A partition-based approach. The VLDB Journal, 2018, 27(1): 53-78

ZHAO Xiang, Ph. D., associate professor. His research interests include large-scale data management and big graph data analytics, knowledge graph.

SHANG Hai-Chuan, Ph. D., specially-designated research associate. His research interests include big data computing and block-chain technologies.

Background

Skyline processing is of great importance in multi-criteria decision-making, which has been a focused theme in the area of data management and big data analytics. In the big data era, massive data request scalable data processing solutions, and parallel Skyline processing emerges. The basic framework of parallel Skyline processing consists of (1) partition points; (2) evaluate local Skyline points in each partition; and (3) merge local Skylines into a global Skyline. Data partitioning plays a key role in efficient Skyline processing. In this research, we investigate a permutation-based partitioning strategy that not only effectively balances workload across multiple partitions as in grid-based partitioning, but also generates fewer local Skyline points compared against state-of-the-art algorithms. Extensive experiments verify the effectiveness of the proposed techniques.

Potential applications of the proposed techniques include massive transactional data analytics and decision-making support, etc.

This work was partially supported by the NSFC under Grant Nos. 61872446, U19B2024 and the NSF of Hunan Province under Grant No. 2019JJ20024.