DFTS:面向大数据集的Top-k Skyline查询算法

魏 亮[1] 林子雨[1] 赖永炫[2,3]
(厦门大学信息科学与技术学院 福建 厦门 361005)[1] (厦门大学软件学院 福建 厦门 361005)[2]
(厦门大学深圳研究院 广东 深圳 518000)[3]

摘 要 Top-k Skyline查询结合了Top-k与Skyline的特性,可以在数据集中找到最好的点。但是,现有的算法在大数据环境下具有较高的时间开销。文中提出一种新的算法DFTS,其可以高效地在大数据集中进行Top-k Skyline查询。DFTS包括3个步骤:首先,利用度值评价函数对数据集进行排序,快速过滤掉大量的点,仅保留足够少的候选集;然后,对候选集进行Skyline查询计算,进一步排除掉Skyline集合外的点;最后,筛选出Top-k的数据点作为最终结果。通过这种方式,DFTS有效减少了算法的运行时间。从理论上证明了DFTS查询的最终结果符合Top-k Skyline查询的要求。基于大数据集的大量实验表明,DFTS具有比现有算法更好的性能。

关键词 Skyline,Top-k,Apache Spark
中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2019.05.023

DFTS:A Top-k Skyline Query for Large Datasets

WEI Liang[1] LIN Zi-yu[1] LAI Yong-xuan[2,3]
(School of Information Science and Engineering,Xiamen University,Xiamen,Fujian 361005,China)[1]
(School of Software,Xiamen University,Xiamen,Fujian 361005,China)[2]
(Shenzhen Research Institute,Xiamen University,Shenzhen,Guangdong 518000,China)[3]

Abstract Top-k Skyline query combines the features of Top-k query and Skyline,which can find the best object in the datasets. However,the available methods can not fit to large datasets well. An efficient Top-k Skyline query method called DFTS was proposed,which can perform well for large datasets. DFTS involves three steps. Firstly,the degree-score function is used to rank the dataset,and a large quantity of objects with low ranking will be filtered out. Secondly,DFTS makes a Skyline query upon the candidates and generates a Skyline subset. Finally,top-k objects with high ranking will be selected from the Skyline subset as the final result. Through these steps,DFTS can significantly reduce the time cost. It is proved that the results of DFTS satisfy the demand of Top-k Skyline query. Extensive experimental results show that DFTS can achieve much better performance for large datasets than state-of-the-art methods.

Keywords Skyline,Top-k,Apache Spark

1 引言

Skyline查询[1]可以帮助用户从数据集的子集中快速地做出选择。近年来,Skyline查询一直是学术界研究的热点问题,研究人员提出了很多代表性的算法,如k-支配Skyline查询[2]、空间Skyline查询[3]、Top-k支配查询[4]以及多层次Skyline[5-6]等。

Skyline查询可以从数据集中找出最好的点。例如,表1中列出5个酒店的信息,酒店的两个属性分别是住宿价格以及酒店到景点的距离。通常情况下,用户更愿意选择价格更便宜以及距离更近的酒店。假设以此为前提条件,那么参照表1中的数据,因为酒店A到景点的距离与酒店B相同,同时A的价格优于B,所以认为酒店A比酒店B更好。可以把这种比较方式定义为"酒店A支配酒店B",记作$A \prec B$。Skyline查询最终就是返回那些不会被其他点所支配的点。因此,对表1进行Skyline查询的结果是:酒店A,D和E。

表1 景点周围的酒店

Table 1 Hotels around the tourist attraction

酒店	价格/元	距离/m
A	100	600
B	120	600
C	300	200
D	210	100
E	150	500

然而,当数据集规模很大时,Skyline查询可能会返回一个很大的子集作为结果,这就意味着用户将会看到一个较大

到稿日期:2018-07-12 返修日期:2018-09-15 本文受国家自然科学基金(61672441),深圳市基础研究计划(JCYJ20170818141325209)资助。
魏 亮(1992-),男,硕士生,主要研究领域为数据挖掘、大数据等;林子雨(1978-),男,CCF会员,主要研究领域为数据库、数据挖掘、大数据等,E-mail:ziyulin@xmu.edu.cn(通信作者);赖永炫(1981-),男,副教授,CCF会员,主要研究领域为数据库、传感器网络数据管理等。

规模的 Skyline 集合。例如,图 1 中的连续曲线部分是函数 $Y=\frac{1}{X}$ 在第一象限的曲线。假设(0,0)为最优属性,即数据越小越好。这里假设有一个数据集 D_1,它的分布为图 1 中的所有点(包含 $Y=\frac{1}{X}$ 曲线上的点)。那么,曲线 $Y=\frac{1}{X}$ 就是数据集 D_1 的 Skyline 集合的有效分布范围。因为一条线上的点是无限的,所以可以给出一个合理的推测:如果 D_1 的数据集足够大,那么 D_1 分布在这条曲线上的点,即 D_1 的 Skyline 集合规模,也将变得很大。进一步地,若将数据集的维度增加到 3 维,假设 $0<x,y,z\leqslant 1$,且 $x^2+y^2+z^2=1$,这是一个球面的一部分,记为 f,如果有一个数据集 D_2 分布在 f 所包裹的这个部分球体上,并且前提条件是认为数据越大越好,则 f 将是数据集 D_2 的 Skyline 集合的有效分布范围。从一条线扩展成一个面,可以认为 Skyline 集合的规模增大了。综上所述,随着数据集规模的变大以及维度的增加,Skyline 的集合也会变大,当 Skyline 集合达到一定规模时,就会返回过多的结果给用户,从而影响用户对结果的理解。

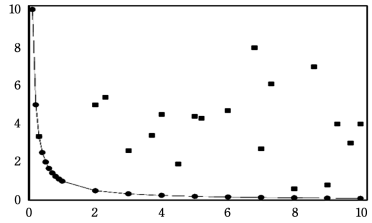

图 1　大数据集中的 Skyline 集合示例

Fig. 1　Example of Skyline in large dataset

Top-k 查询可以解决 Skyline 集合过大的问题。Man 等[4]把 Skyline 查询与 Top-k 查询结合在一起,提出了 Top-k 支配查询,记为 TKDQ。通常,Top-k 查询需要定义一个评分函数来评估每个数据点。TKDQ 的评分函数需要获得数据集中点与点之间的支配关系,这个过程需要大量的时间代价,因为计算点与点之间的支配关系的时间复杂度是 $O(N^2)$ 级别的。因此,TKDQ 在大数据集下十分低效。Endres 等[5-6]通过结合 Top-k 与 Skyline 查询提出了多级网格 Top-k Skyline 算法,但是该算法需要根据数据集生成相应的网格结构,当数据量很大时,构建网格数据结构就变得相当困难,需要较高的时间开销以及大量的空间开销。因此,该算法既增加了时间开销,又不适用于数据取值个数较多的情形。

本文提出了一种新的 Top-k Skyline 算法——基于度值过滤的 Top-k Skyline 查询(Degree Based Filter Top-k Skyline Query,DFTS)。DFTS 可以从大数据集中高效地找到最好的数据点,其实现分为 3 个部分。首先,设计度值评分函数来对数据集中的所有数据点进行排序,并且根据排序的高低过滤掉大量的数据点,仅保留少量的数据点作为候选集。然后,对候选集进行 Skyline 查询,排除掉那些在候选集中被其他点所支配的数据点。最后,按排序顺序输出最终的 Top-k 结果。

总体而言,本文的主要贡献如下:

(1)提出了一种新的 Top-k Skyline 查询算法 DFTS,可以高效地从大数据集中找到最好的点;

(2)提出了度值评分函数,用来帮助 DFTS 对数据集进行快速的排序过滤;

(3)通过大量实验证明了 DFTS 在大数据环境下的高效性。

本文第 2 节介绍了 Top-k 支配查询算法以及多层次网格 Top-k Skyline 算法,并讨论了其不足之处;第 3 节详细介绍了 DFTS 算法;第 4 节给出了相关的实验结果;第 5 节介绍相关工作;最后总结全文。

2　Top-k Skyline 查询的相关算法

本节将探讨已有的 Top-k Skyline 查询的相关算法,包括 Top-k 支配查询算法和多级网格 Top-k Skyline 查询算法。

2.1　Top-k 支配查询算法

Top-k 支配查询算法 TKDQ 是一种结合了 Top-k 查询和 Skyline 查询的算法。TKDQ 的具体流程如图 2 所示。

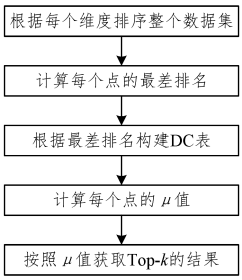

图 2　TKDQ 流程图

Fig. 2　Flowchart of TKDQ

2.1.1　TKDQ 的详细描述

Skyline 查询可以从数据集中找到最好的所有点。Skyline 查询的结果是数据集的一个子集,并且这个子集中的每个成员数据点都不会被原数据集中的任意一个点支配。下面给出两个基础的定义。

定义 1(支配)　给定一个 m 维的数据集 DS,以及数据点 $O_p,O_q\in DS$。$X_{p,i}$ 表示数据点 Op 的第 i 维的数据,如果 $X_{p,i}\geqslant X_{q,i}(1\leqslant i\leqslant m)$,并且 $\exists X_{p,j}>X_{q,j}(1\leqslant j\leqslant m)$,则认为 O_p 支配 O_q,记作 $O_p\prec O_q$。这里假设数据值越大越好。

定义 2(Skyline)　给定一个 m 维的数据集 DS,Skyline 是 DS 的一个子集,在该子集中所有数据点都不会被 DS 中的任意数据点支配。

Top-k 查询是从数据集里找到几个最好的点。通常,用户需要设定一个评价函数来对数据集中的点进行排序。TKDQ 的评价函数如下:

$$\mu(O)=|\{O'\in DS|O\prec O'\}|$$

评价函数 $\mu(O)$ 将返回在数据集中被点 O 所支配的点的数量。因此,Top-k 支配查询可以从数据集中返回 μ 值最高的几个数据点。在此基础上,Siddique 等[7]利用这个算法来处理大数据问题,具体过程如图 2 所示。首先,对数据集的每个维度进行排序,得出每一个点的最坏排名。其次,根据最坏排名构建 DC 集合,该集合包含每个数据点之间的支配关系。然后,参照 DC 集合进行计算来得出每个数据点的 μ 值。最后,根据 μ 值对整个数据集合进行排序。

2.1.2　TKDQ 的不足

μ 值评分函数存在一些问题。从表 2 可以看出,$\mu(A)=$

3，$\mu(E)=0$。根据 μ 值评价函数，A 比 E 好。然而，从数值角度来看，在价格上 E 比 A 仅仅贵了 20%，但是在距离上 A 却是 E 的数倍。因此从实际情况来看，E 可能更受青睐，从而可以合理地推断出 TKDQ 很可能会将一些很好的数据点排除在查询结果外。

表 2 景点周围的酒店信息

Table 2 Hotels around the tourist attraction

酒店	价格/元	距离/m
A	100	500
B	110	650
C	110	700
D	110	600
E	120	100
F	500	20

此外，TKDQ 在大数据环境下是十分低效的。为了计算每一个数据点的 μ 值，首先要求出最坏排名，且每个维度都要统计，那么一个维度最快的排序算法的时间复杂度为 $O(n\log n)$，如果有 m 维，就需要 m 次排序。然后根据数据点的最差排名生成 DC 表（见表 3），此过程花费的时间相对较少，可以忽略不计。DC 表的内容是比数据点在最差排名维度更差的点的集合。最后，按照 DC 表计算 μ 值。根据表 3，要计算 $O_{1,1}$ 的 μ 值，只需要计算表中 $O_{1,1}$ 的支配候选集中的点是否会被 $O_{1,1}$ 支配，那么影响整个算法时间开销的因素就是 DC 表中包含的候选项的数目 M，因此，算法的时间复杂度为 $O(M)$。根据文献[7]的实验结果，TKDQ 在正相关的数据中表现最好，而对于正相关的数据集而言，数据会集中分布在均值附近，呈线性分布。对于在均值附近的点，其最差排名可以估值为 $\frac{N}{2}$，所以，每个在均值附近的点都拥有 $\frac{N}{2}$ 左右的候选项；头尾两端可以平均为 $\frac{N}{2}$。因此，总的候选集 $M\approx\frac{N^2}{2}$，随着 N 的增大，M 的取值会非常大，从而导致 TKDQ 的时间开销非常高，无法适用于大数据集的情形。

表 3 DC 表示例

Table 3 Example of DC set

序号	支配候选集
$O_{1,1}$	$\{O_{3,2}\}$
$O_{1,2}$	$\{O_{2,2},O_{3,2}\}$
$O_{2,1}$	$\{O_{2,2},O_{3,2}\}$
$O_{2,2}$	$\{O_{3,2}\}$
$O_{3,1}$	Null
$O_{3,2}$	Null
$O_{3,3}$	$\{O_{2,1},O_{2,2},O_{3,2}\}$

2.2 多级网格 Top-k Skyline 查询算法

多级网格 Top-k Skyline 查询算法[5-6]必须要有一个网格的数据结构，这个网格结构包含了各个数据点间的直接关系。如图 3 所示，其中白底的节点都是没有具体元素的，S_{ml}^i 代表第 i 层的 Skyline 集合。第一层（即 S_{ml}^0）是完整数据集的 Skyline 集合；当数据集去掉第一层后，第二层就是剩余数据集的 Skyline 集合，依此类推。这个数据结构被存储在内存中，以加快查询的速度。所要求取的 Top-k Skyline 结果就是遍历这个结构产生的。

这种网格的形状更适用于数据取值个数较少的数据集，比如酒店的星级仅包括 0 星到 5 星，这时整个图的深度不会太大。但是，对于取值个数较多的数据，比如 $0.001\leqslant x\leqslant 10000$，如果利用网格结构，就会产生十分可观的结构深度。此外，在大数据集中建立网格结构也将耗费大量的时间。而对于非特殊的数据集来说，数据的取值个数并不一定是较少的，因此这种算法对于大数据集的 Top-k Skyline 查询也不是一种较好的解决方案。

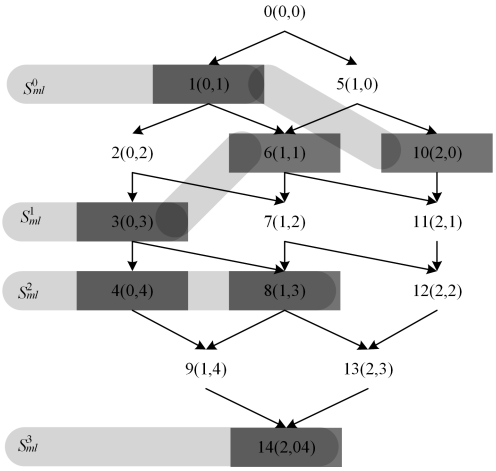

图 3 多层级网格 Top-k Skyline 示例

Fig. 3 Example of multi-level lattice Top-k Skyline

3 DFTS 算法

为了对大数据集进行高效的 Top-k Skyline 查询，本文提出一种新的基于度值过滤的 Top-k Skyline 查询算法 DFTS。通过基于度值进行快速过滤的方法，DFTS 大幅减小了需要进行 Skyline 查询的候选集的规模，从而达到高效地获取结果的目的。

3.1 DFTS 算法概览

DFTS 算法主要包含 3 个步骤：首先，根据每个数据点的度值评价函数得分进行排序，并根据排序快速地过滤掉大量的数据点，仅仅保留足够少量的数据点作为候选集（见 3.2.2 节）；然后，对候选集进行 Skyline 查询计算（见 3.3 节）；最后，从两步筛选后的集合中获取 Top-k 的点作为最终结果（见 3.4 节）。DFTS 算法的具体流程如图 4 所示。

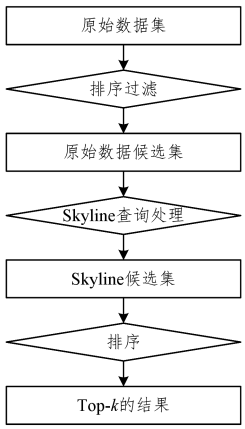

图 4 DFTS 流程图

Fig. 4 Flowchart of DFTS

3.2　排序与过滤

本节首先对度值评价函数进行介绍，并利用简单的示例来解释该函数过滤数据集的工作机理。

3.2.1　根据度值评价函数进行排序

Z-值标准化是统计学中的一种方法。Z-值可以用来衡量某个数据距离数据集平均值的距离，因此其可以很好地衡量数据在数据集中地位的高低。这里给出 Z-值标准化的计算公式：$X^* = \frac{X-\mu}{\sigma}$。其中，$\mu$ 是均值，σ 是标准差，$\sigma=\sqrt{\sum_{i=1}^{n}(X_i-\mu)^2}$。但是，如果直接使用 Z-值作为衡量指标，会存在问题。例如，基于表 4 可以很快得出两列数据的标准差是一样的，对于 A 点，计算出的第一维度为 $X_1^* = -\frac{\sqrt{2}}{2}$，第二维度为 $X_2^* = -\frac{\sqrt{2}}{2}$，这两个值是相等的。即如果直接使用 Z-值作为评价函数，会很难区分两列不同数值之间的优劣。假设数值越低越好，则 90 相比于均值 100 的优秀程度为 10%，同时 9990 相比于均值 10000 的优秀程度只有 0.1%。而 Z-值的判定结果认为 90 与 9 990 在各自列的地位是一样的。因此，用 Z-值作为多维数据的评价函数还存在一定的不足。

表 4　Z-值计算示例

Table 4　Example of Z-Value

酒店	价格/元	距离/m
A	90	9990
B	100	10000
C	110	10010

定义 3（度值）　给定一个 m 维数据集 DS，以及数据点 $O_i \in DS$，O_i 的第 j 维度的度值为 $\tau_{i,j}$，则 $\tau_{i,j} = \frac{X_{i,j}-\mu_j}{\mu_j}$。其中，$X_{i,j}$ 代表数据点 O_i 的第 j 维度的数值，μ_j 代表第 j 维度的均值，$\mu_j = \frac{\sum_{i=1}^{n} X_{i,j}}{n}$。

以表 4 为例，可以计算得到 $\tau_{1,1} = -0.1$，$\tau_{1,2} = -0.001$，这两个度值可以很明显地展示出数据在各自维度的地位。

度值用来评估特定维度数值的重要性。通常，一个数据集会有很多维度，把数据点每一个维度的度值进行求和后作为这个数据点的最后得分。称该评价函数为度值评价函数，其相关定义如下。

定义 4（度值评价函数）　各维度的度值之和被称为数据点 O_i 的得分，记作 $\phi(O_i)$，则 $\phi(O_i) = \sum_{j=1}^{m} \tau_{i,j}$，其中，$m$ 表示维度数量。

根据定义 3，$\tau_{i,j}$ 可以为负值。为了便于理解，这里假设数值越大越好（后面的内容都满足这个假设）。如果 $\tau_{i,j}$ 为正值，那么 $X_{i,j}$ 就比第 j 列的平均值要好；反之则反。由于 $\phi(O_i)$ 是 τ 值的和，因此 $\phi(O_i)$ 也可以是负值。当 $\phi(O_i)$ 为负值时，则认为 O_i 比数据集的平均水平差。在计算 τ 值时，可以不用考虑数据单位不一致的因素，直接累加 τ 值，使得不同列的数据具有可比性。

3.2.2　根据度值评价函数进行过滤

根据上述度值函数的定义，可以对数据点进行排序，并且按照排序顺序快速地过滤掉大量的数据点，仅保留足够少的数据点作为候选集。下面介绍度值评价函数在过滤过程中的工作原理。

定理 1　给定一个 m 维的数据集 DS，以及 3 个数据点 $O_p, O_q, O_k (O_p, O_q, O_k \in DS)$，如果 $O_p \prec O_q$ 并且 $O_q \prec O_k$，那么 $O_p \prec O_k$。

上述定理显然成立，这里不给出相关证明。

引理 1　给定一个 m 维的数据集 DS（数据点 $O_i \in DS$），以及 DS 的 Skyline 集合 SL。如果 $O_i \notin SL$，则 $\exists O_j \in SL$，$O_j \prec O_i$。

证明：假设 $O_i \notin SL$，那么必然有 $\exists O_j \prec O_i$。如果 $O_j \notin SL$，则 $\exists O_k \prec O_j$。同时根据引理 1，支配是具有传递性的，所以 $O_k \prec O_i$。因此，若 $O_k \notin SL$，则传递性无限延续，显然不存在无限大的数据集，所以引理 2 成立。

定理 2　给定数据集 DS 的 Skyline 集合 SL，则 $\exists O_i \in SL$，且 $X_{i,j}$ 是第 j 维度的最大值。

证明：假设 $O_j \in DS$，且 $X_{j,k}$ 为 k 维度的最大值，那么根据引理 1，若 $O_j \notin SL$，则 $\exists O_p \in SL$，$O_p \prec O_j$，这就意味着 $X_{j,k} \leqslant X_{p,k}$，得出 $X_{p,k}$ 就是最大值，因此定理 2 成立。若 $O_j \in SL$，则 $X_{j,k}$ 即为最大值，定理 2 成立。

引理 2　如果 $\phi(O_i) > \phi(O_j)$，则 $O_j \not\prec O_i$，表示 O_j 不能支配 O_i。

证明：假设 $O_j \prec O_i$ 并且 $\phi(O_i) > \phi(O_j)$。因为 $O_j \prec O_i$，所以 $\forall X_{j,q} \geqslant X_{i,q}$，且 $\exists X_{j,p} > X_{i,p}$。那么 $\phi(O_j) > \phi(O_i)$。假设不成立，即引理 2 成立。

根据前文的描述，数据点的 ϕ 值是由各个维度的 τ 值进行累加的，因此各个维度的不同优劣标准会对引理 2 产生影响。比如，假设存在一个数据集（80，90，100，110，120），其均值为 100，每个数据点的值表示负债数目，因此值越小越好。然而，对于数据点 80 而言，可以计算得到 $\tau = -0.2$，对于数据点 120 而言，$\tau = 0.2$，计算结果显示 80 反而不如 120 好，这违背了数据的真实含义。为了让 80 可以正常地优于 120，只需要将这个维度的 τ-值乘上 -1 的权重。对于普通数据集而言，初始化一个权重向量是比较容易的。实现上，可以建立一个 m 长度的数组，m 表示数据集的维度数，而数组元素则是 1 或 -1 权重。1 表示数据越大越好，-1 表示数据越小越好。这样在实际应用中，即使每个维度的优劣标准不一，也可以保证引理 2 依然成立。

在解决优劣标准不一的问题之后，就可以为各个数据点计算 ϕ 值得分；然后依据 ϕ 值得分进行排序，并将排序靠后的数据点过滤掉，保留少数的数据点作为候选集；接着对候选集进行 Skyline 查询计算，从候选集取出的 Skyline 集合记为 CSL，整个数据集的 Skyline 集合记为 SL。可以从理论上证明 $CSS \in SL$。下面给出相关定理及其证明。

定理 3　给定一个 m 维的数据集 DS，以及 DS 的 Skyline 集合 SL。保留 ϕ 值得分高的数据点作为候选集 CS，根据 CS 可以计算得到它的 Skyline 集合 CSL，则 $CSL \in SL$。

证明：因为 $\forall \phi(O_i) \geqslant \forall \phi(O_j)$，其中，$O_i \in CS$，$O_j \in$

$\{DS-CS\}$，根据 Skyline 引理 2 可得，$O_j \prec O_i$。此外，根据 Skyline 的定义，$\exists O_k \in CS, O_k \prec O_i$，其中，$O_i \in CSL$。$O_j$ 和 O_k 合并后即为整个数据集 DS，得 $\exists O_j \in DS, O_j \prec O_i$，其中，$O_i \in CSL$。因此 CSL 满足 Skyline 定义，即 $CSL \in SL$，定理 3 得证。

根据定理 3，对于数据集 DS，按照 ϕ-值的排名可以过滤掉大量排名靠后的数据点。然后，根据定理 2，数据集的 Skyline 集合中必定包含每一个维度的极值点。需要注意：极值点可能没有较高的 ϕ-值，比如对于一个均值为(100,100)的集合，其中一个极值点为(150,1)，以数值越大越好作为前提条件，则该极值点的 $\phi((150,1))=-0.49$，该极值点在较大概率上是不会被选进候选集的。但是，极值点对于数据集来说具有较大的价值，因此可以将极值点作为特殊考虑范畴。计算过程中将保留每一个维度的所有极值点中 ϕ-值最高的极值点，因此共有 m 个极值点被保留(可重复)，m 表示维度数，而且根据引理 2，m 个极值点都属于 DS 的 Skyline 集合。算法 1 展示了整个过滤算法的具体过程。

算法 1 过滤算法 Filtering(DS)

```
Input:Dataset DS
Output:candidates CS,maximum set Mset
begin
initial weight vector:W
  //计算 μ 向量
    calculate the means vector:Savg
    for each Oi∈DS do
      //计算数据点的 ϕ-值
        for each Xi,j∈Oi do
            τi,j=(Xi,j−Savgj)/Savgj×Wj
        end for
            ϕ(Oi)=Σ_{j=1}^{m} τi,j
      //维护 m 个极值点集合:Mset
        for each Xi,j∈Oi do
          if (Xi,j≥maxj and ϕ(Oi)≥ϕ(maxj))
            update the Mset
          end if
        end for
    end for
    //取出 Top-k ϕ-值的数据点
    CS = DS.top(k)
end
```

3.3 Skyline 计算

经过滤算法处理后，将得到两个小的数据集，即候选集合 CS 和极值点集合 $Mset$。但是，$Mset$ 不需要与 CS 进行合并，因为根据定理 3，CS 的数据点都不会被 DS 中 ϕ-值较小的点所支配，所以如果 $Mset$ 的数据点的 ϕ-值较大，就会出现在 CS 中；反之，就不会支配较大的点。因此，最后只需要计算 CS 的 Skyline 集合 CSL。由于经过了过滤后候选集的规模已大幅缩减，因此在对 CS 进行 Skyline 查询时，可以采用已有的计算方法。本文的研究重心在于过滤方式，所以，在实现 CSL 的计算时直接采用双重循环的方式，即两两数据进行比对。

3.4 获取 Top-k 的数据点

尽管 CSL 是过滤后的结果，但是对于 Skyline 中的数据点而言，它们大概率地拥有较大的 ϕ-值，所以 CS 包含的大部分数据点都可能进入 CSL。因此，仍然需要对 CSL 进行排序，选出 Top-k 个数据点作为输出。由于 CSL 中的数据点已经不会被其他点所支配，因此继续依据 ϕ-值对 CSL 进行排序输出。最后，即选出原始数据集 DS 的 Skyline 集合中的 Top-k 个结果。

$Mset$ 将以一个表的形式输出给用户。$Mset$ 的一个作用是作为 Top-k Skyline 结果的参照，以便于观察是否选取到极值点；另一个作用是可以满足用户的特殊需求，比如用户资金不足，只能选取最便宜的酒店，而不用考虑酒店环境等。

3.5 性能分析

根据上面的介绍，DFTS 是十分高效的 Top-k Skyline 的计算方法。下面将从理论层面给出 DFTS 算法的性能分析。

给定一个 M 维的数据集，其中包含 N 个数据点。首先需要求出 μ 向量，该过程的时间复杂度为 $O(N)$。然后计算各个数据点的 ϕ-值，这个过程需要遍历整个数据集，计算每个数据点各维度的 τ-值，并累加求和，该过程的时间复杂度为 $O((M+1)N)$。其次对加入了 ϕ-值的 DS 进行 top 操作。top 函数是 Spark 实现的，它采用的方法是维护一个 $2K$ 的容量缓存(K 为 CS 规模的大小)，每当容量满时找到中值点，根据中值将数据分为两部分，并保留需要的部分，即把 $2K$ 重新变成 K，遍历完数据集后再做一次排序，整个过程的时间复杂度为 $O(N+K\log K)$。接着对 Top-K 结果进行 Skyline 查询，时间复杂度为 $O(K^2)$。最后取 Top-k，k 为需要的最终结果数量。由于 M，K 和 k 都是远小于 N 的，因此可以认为 DFTS 是一个 $O(CN)$ 量级的算法，其中，C 为常数。

4 实验设计与结果分析

本文开展了大量实验来证明 DFTS 算法的高性能。算法代码采用 Scala 语言编写，在 Spark 集群中运行，集群由 4 台 Ubuntu16.04 机器搭建，每台机器配备 Intel(R) Core(TM) i3 3.10 GHz 的处理器以及 4 GB 的内存，机器之间通过路由器相连。每种实验重复进行 10 次，去掉最好与最差结果，然后取平均值作为最终结果。实验使用的数据集由人工合成，实验假设数据值越大越好，生成的数据有 3 种分布：正相关分布、反相关分布以及随机分布。

首先对一组 2－6 维，以及拥有 3 种不同数据分布的含有 100 万个数据点的数据集进行实验。3 种分布分别是：正相关、反相关以及无关。这个实验的目的是探究数据集的维数以及数据集的数据分布对 DFTS 算法性能的影响。然后，为了找出数据量对算法性能的影响，分别对 10 万、100 万、500 万、1000 万的正相关数据集进行实验，这里只对正相关数据进行实验，因为上文已经证明算法不受数据分布的影响。此外，还将 DFTS 与 TKDQ 进行对比实验来验证其高效性。但是，本文没有将 DFTS 与多层级网格 Top-k Skyline 查询算法进行对比，因为两者适用的情况不同，DFTS 是对原始数据直接进行操作，而多层级网格 Top-k Skyline 查询算法需要构建网格数据结构，在数据集规模很大的情况下，在内存存储一个

巨大的数据结构并不容易实现。此外,实验结果也表明,在数据集规模较小的情况下,DFTS 的性能收益远不如大规模数据集。

4.1　评估 DFTS 算法性能的影响因素

如图 5 所示,在不同的数据维度下,对于 3 种不同的数据分布类型而言,DFTS 算法的运行时间基本一致,这意味着数据分布类型不会对 DFTS 的运行时间产生明显影响。

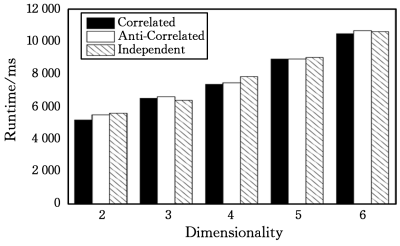

图 5　维度与数据分布对 DFTS 运行效率的影响

Fig. 5　Influence of dimension and distribution on efficiency of DFTS

从图 5 中还可以看到,当数据集为 2 维时,DFTS 的运行时间大概在 5400 ms 左右;同时,随着维数从 2 维依次增加到 6 维,程序的运行时间分别为 6 500 ms,7 500 ms,9000 ms,10600 ms 左右。实验结果表明,在数据集维数增加时,DFTS 的运行时间几乎以一个平稳的趋势增加,不会导致算法性能的急剧下降。这个实验结果与第 3.5 节关于 DFTS 算法的性能分析结果相符,即 DFTS 的运行时间与数据集的维度数量之间存在线性关系。

本文在不同数据量的数据集下构建了相关的对比实验。如图 6 所示,运行时间与数据量之间的关系也是线性的。正如前文所述,DFTS 的时间复杂度是 O(N)级别的,因此数据量是 DFTS 运行时间的另一个主要影响因素。图 6 中的线条几乎是一条直线,斜率约为 0.08,各点的运行时间分别是 3072 ms,10861 ms,42981 ms 和 85298 ms。

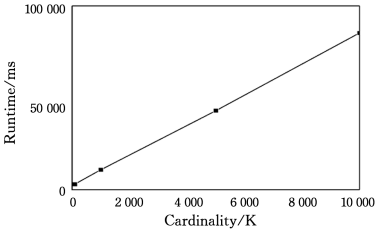

图 6　数据量对 DFTS 运行效率的影响

Fig. 6　Influence of cardinality on efficiency of DFTS

4.2　与 TKDQ 的对比

本文构建了相关的实验来对比 DFTS 和 TKDQ 算法的性能,实验结果如图 7 所示。图中有两条直线,TKDQ 这条线上的点代表 TKDQ 在 7.5 万、15 万、22.5 万以及 30 万的数据量下的运行时间;而 DFTS 的这条线代表 DFTS 在 100 万数据量下的运行时间。数据集都是 6 维正相关分布,这里采用正相关的数据的原因是根据文献[7]的实验结果,TKDQ 在正相关数据分布中表现最优秀。DFTS 明显可以用更短的时间处理更大量级的数据集。

如图 7 所示,DFTS 只用了大约 10 s 来处理 100 万的数据量,而 TKDQ 则使用了 370 s 左右的时间来处理仅 30 万的数据量。该实验结果表明,DFTS 具有比 TKDQ 更加高效的大数据处理能力。

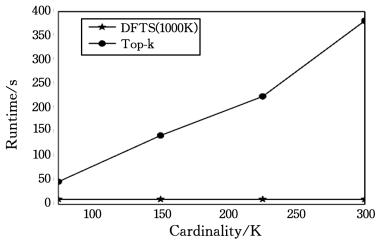

图 7　DFTS 与 TKDQ 的对比

Fig. 7　Comparison between DFTS and TKDQ

4.3　DFTS 在不同规模数据集中的收益分析

为了评价算法收益的大小,这里给出一个评价收益的标准:算法在单位时间内所能处理的数据量(记为 I-值)。如表 5 所列,数据集规模为 1 k 时,DFTS 算法的 I-值为0.00097,表示平均每 1 ms 可以处理 0.97 个数据点;当数据规模分别是 10 k,50 k,100 k,500 k,1000 k,5000 k 以及 10000 k 时,相对应的 I-值分别为 0.00799,0.02946,0.04684,0.11371,0.13845,0.18104,0.18927。从实验结果可以看出,DFTS 算法在数据集规模较小的情况下,收益远不如大规模数据集下的收益;而且随着数据规模的增大,DFTS 算法的收益也会增加。因此,DFTS 算法非常适用于大规模数据的情形。

表 5　DFTS 在不同规模数据集中的收益

Table 5　Profit of DFTS on different scale datasets

Cardinality/k	Runtime/ms	I-值
1	1035	0.00097
10	1251	0.00799
50	1697	0.02946
100	2135	0.04684
500	4397	0.11371
1000	7223	0.13845
5000	27617	0.18104
10000	52834	0.18927

5　相关工作

自 Borzsonyi 等[1]提出 Skyline 查询操作之后,Skyline 查询便得到了广泛研究。很多优秀的算法被学者提出,用于解决 Skyline 查询问题。

Borzsonyi 等[1]为了高效地计算数据集的 Skyline 集合,提出了 BNL 算法、分而治之算法以及基于 B 树的图方法。其中,分而治之算法把数据集切分成几个部分,使得每个部分都能够存在内存之中。这个方法的可行性意味着 Skyline 查询操作是可以在分布式环境下进行的。随后,Chomicki 等[8]提出了一种预排序的算法——Sort-Filter-Skyline(SFS),其为 BNL 的一个变种。之后,Godfrey 等[9]提出了 LESS(Linear Elimination Sort for Skyline)。LESS 可以在预处理阶段排除不属于集合的数据点,因此进一步提高了 SFS 的性能。Lee 等[10-11]提出了 Z-SKY 框架以及一系列的算法,如 Z 搜索、Z 插入、Z 删除、Z 修改等。Chaudhuri 等[12]提出了鲁棒技术,并且利用日志抽样方法(LS)来评估数据点数量以及查询的消耗。Zhang 等[13]提出了基于内核的算法(KB),并且把 LS 和

KB 应用到 K-支配 Skyline 查询。KB 算法是一种可以在高维数据集中取得优异表现的算法。

有一些研究采用了基于索引的算法。Tan 等[14]提出了位图方法;Kossmann 等[15]提出了最近邻算法,该算法的主要思想就是利用最近邻对数据集进行递归分割;Papadias 等[16-17]则改善了最近邻算法,提出了分支限界 Skyline 算法(BBS)。

后来的一些研究对 Top-k 算法与 Skyline 算法进行了有效结合。其中,Endres 等[5-6]提出的多级网格 Top-k Skyline 算法,为 Top-k Skyline 查询问题提供了一种高效的查询算法。而在大数据时代,分布式的 Top-k Skyline 查询问题存在着较高的研究价值。例如,Man 等[4]提出了一种 Skyline 查询的变种——Top-k 支配查询,用来从数据集中获取最好的数据点;Siddique 等[7]将 Top-k 支配查询与 K-Skyband 结合后应用在分布式环境中,用来处理大数据问题。

结束语 为了从大数据集中高效地获取最具代表性的数据点,本文提出了 DFTS 算法。该算法采用了一种新的称为度值的评估方法,用来评估数据集中数据点的各维度数值的重要性,并且利用度值来构建新的评估函数(称为度值评估函数)。通过这个新的函数,DFTS 可以快速地过滤掉大量评分低的数据点,仅保留足够少的具有高评分的数据点作为候选集,从而大幅缩减候选集的规模,因此可以大大减少算法的时间开销。此外,本文还开展了大量实验,实验结果表明,相比于现有处理大数据集合的 Top-k Skyline 查询算法,DFTS 具有更好的性能。

参 考 文 献

[1] BÖRZSÖNYI S,KOSSMANN D,STOCKER K. The Skyline operator[J]. Data Engineering,2001,1:433-442.

[2] CHAN C Y,JAGADISH H V,TAN K L,et al. Finding k-dominant skylines in high dimensional space[C]// Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data. Chicago:ACM,2006:503-514.

[3] SHARIFZADEH M,SHAHABI C. The spatial skyline queries [C]// Proceedings of the 32nd International Conference on Very Large Data Bases. Seoul,Korea:VLDB Endowment,2006:751-762.

[4] MAN L Y,MAMOULIS N. Efficient processing of top-k dominating queries on multi-dimensional data[C]// Proceedings of the 33rd International Conference on Very Large Data Bases. Vienna:VLDB Endowment,2007:483-494.

[5] ENDRES M,PREISINGER T. Behind the skyline [J]. Proceedings of DBKDA,2015,15.

[6] PREISINGER T,ENDRES M. Looking for the best,but not too many of them:multi-level and top-k skylines[J]. Int. J. Adv. Softw,2015,8(3):4.

[7] SIDDIQUE M A,TIAN H,MORIMOTO Y. Selecting Representative Objects from Large Database by Using K-Skyband and Top-k Dominating Queries in MapReduce Environment[M]// Advanced Data Mining and Applications. Springer International Publishing,2014:560-572.

[8] CHOMICKI J,GODFREY P,GRYZ J,et al. Skyline with presorting:Theory and optimizations[M]// Intelligent Information Processing and Web Mining. Springer,Berlin,Heidelberg,2005:595-604.

[9] GODFREY P,SHIPLEY R,GRYZ J. Maximal vector computation in large data sets[C]// Proceedings of the 31st International Conference on Very Large Data Bases. Trondheim:VLDB Endowment,2005:229-240.

[10] LEE K C K,ZHENG B,LI H,et al. Approaching the skyline in Z order[C]// Proceedings of the 33rd International Conference on Very Large Data Bases. Vienna:VLDB Endowment,2007:279-290.

[11] LEE K C K,LEE W C,ZHENG B,et al. Z-SKY:an efficient skyline query processing framework based on Z-order[J]. The VLDB Journal,2010,19(3):333-362.

[12] CHAUDHURI S,DALVI N,KAUSHIK R. Robust cardinality and cost estimation for skyline operator[C]// 22nd International Conference on Data Engineering (ICDE'06). Atlanta:IEEE,2006:64-64.

[13] ZHANG Z,YANG Y,CAI R,et al. Kernel-based skyline cardinality estimation[C]// Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data. Providence:ACM,2009:509-522.

[14] TAn K L,Eng P K,Ooi B C. Efficient Progressive Skyline Computation[C]// International Conference on Very Large Data Bases. Rome:Morgan Kaufmann Publishers Inc. 2001:301-310.

[15] KOSSMANN D,RAMSAK F,ROST S. Shooting stars in the sky:An online algorithm for skyline queries[C]// Proceedings of the 28th International Conference on Very Large Data Bases. Hong Kong:VLDB Endowment,2002:275-286.

[16] PAPADIAS D,TAO Y,FU G,et al. Progressive skyline computation in database systems[J]. Acm Transactions on Database Systems,2005,30(1):41-82.

[17] PAPADIAS D,TAO Y,FU G,et al. An optimal and progressive algorithm for skyline queries[C]// Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data. San Diego:ACM,2003:467-478.