中国科学技术大学

学士学位论文

题　目　基于预知解的高性能 Top-k 算法

英　文　High Performance Top-k Algorithms Based on Solution Prediction

院　系　计算机科学技术学院

姓　名　陈东渠

学　号　PB08210307

导　师　孙广中　副教授

日　期　2012 年 05 月 15 日

致　谢

在我的毕业设计结题之际，我要向一些对我的大学生活有着深刻影响的人表示感谢。

首先，我要感谢我的导师孙广中教授。孙老师是一位年轻有为的副教授，他提倡学术上的自由，主张思想创新，注重理论和实践的结合。孙老师十分重视培养学生的学术研究能力，对我的科研工作做了细心地指导，带领着我们这个由本科生和研究生共同组成的团队不断地前进。我在科大的两年之内就完成了三篇学术论文，离不开孙老师的指导与帮助。在实验室工作的这两年，是我科研工作的开始，影响了我一生，在此我要对他表示深深的感谢。

其次，我必须感谢我的哥哥陈东鹏，他无疑是整个大学和整个人生中对我帮助最大的人。六年前，是他带我来到科大，是他为我开启了计算机科学的大门。二十年来，他对我的帮助和支持是无尽的，不论是生活上、学习上，还是科研上。即使身在香港科技大学，他依然时刻关心着我的生活和学习，同样身为计算机专业的学生，他用自己宝贵的经验和教训，为我铺平了大学四年的成长道路。他让我学会了如何快乐地生活、高效率地学习和高品质地工作，这些知识都是无价的，我对他的谢意难以言尽。

再次，我要感谢这两年来在学习和科研上不断给予我指导与帮助的师兄们。06 级的龚真强师兄是实验室给予我最多指导的师兄，他愿意在繁重的工作中指导我的学习，为我的 idea 提建议，甚至在微软亚洲研究院熬夜为我修改论文。我在 UC Berkeley 实习期间，龚真强师兄也非常热情地帮助我，是龚真强师兄引领我走进学术的世界，体会到算法的乐趣所在。

感谢和我一起生活、学习的朋友们，与他们一起探讨问题共同进步，是我在科大生活中的一大乐趣。感谢我的班主任老师和我在科大的所有老师，感谢四年来他们对我的谆谆教诲。

最后，我要感谢在我身后默默支持我的亲人们。二十年一路走来，没有他们的支持，我不可能走到这里。他们对我的鼓励，我铭记在心。

目　录

摘　要

Top-k 检索算法研究，是信息检索领域的一个分支研究课题。Top-k 检索在搜索引擎、信息查询、数据库管理等实际应用背景中有较广泛的应用，因此近年来受到较大的关注。本文旨在研究基于预知解的高性能 Top-k 检索算法，即通过对 Top-k 检索的解空间的相关推断，来提高实时 Top-k 检索的效率。本文首先引入 Reverse Top-k 检索（2011 年）的概念。Reverse Top-k 检索在二维情况下，对于给定点，求出使其满足 Top-k 条件的聚合函数的集合，其本质是 Top-k 解空间的预知性问题。基于 Reverse Top-k Algorithm (RTA)的思想，本文提出了基于 RTA 的 Top-k 检索算法 (TBRT)。预处理过程中对所有的数据项都求出 Reverse Top-k 的解，而在实时检索处理的时候进行逐点比较，即可求得 Top-k 解集。随后，本文提出全新的预知解算法 Solution Prediction Algorithm (SPA)，基于最底层的几何意义：聚合函数和数据项均可视为向量，数据项的聚合得分可视为两向量的点乘积。SPA 将 Top-k 的解空间划分为多个保序区域，并通过一系列优化处理将保序区域合并为保解区域。由此构造出了聚合函数的函数空间到 Top-k 解集空间的映射关系，为实时 Top-k 检索带来了很大的效率提升。本文为每个算法作出了正确性的证明并给出了相应的实例。最后，通过与另外三个 Top-k 检索算法 TA（2001 年）、DTA（2011 年）和 S-DTA（2011 年）的比较，可以看出 TBRT 和 SPA 使得 Top-k 检索效率有了较显著的提高。

关键词：算法，信息检索，Top-k 检索，Reverse Top-k 检索，预知解

Abstract

Top-k query has been widely studied recently in many applied fields such as information retrieval, multimedia databases and data mining. In this paper, we focus on high performance top-k query algorithms using prediction methods on solution space. We first introduce the reverse top-k query, a problem proposed recently. Reverse top-k query is run on a two dimensional dataset. Given a specific data item, reverse top-k returns a set of aggregation function where the given item is among the top-k answer when the aggregation function belongs to the set. Actually, reverse top-k query is a work based on solution prediction. Therefore, RTA can evolve into a series of top-k algorithms, named Top-k algorithms Based on Reverse Top-k (TBRT). The main idea of TBRT is to run RTA on every item in the given dataset offline and to find out the top-k answers by test the aggregation function on each item's RTA result. In order to break the limitation on the number of dimensions as well as to improve the efficiency, we propose our second algorithm, the Solution Prediction Algorithm (SPA). SPA is based on a very simple geometric meaning: The aggregation functions and the items can be represented as vectors, so that the aggregating score is the dot product of the function vector and the item vector. SPA introduces the concept of Rank Watershed, which is a hyper-plane crossing the center and perpendicular to the line segment between two items p and q. It can be proved that p and q have the same score for the aggregation functions on their rank watershed. The relative rank of p and q changes when the aggregation functions move from one side to the other side of their rank watershed. Consequently, the rank watersheds partition the solution space into a set of areas. The aggregation functions in an area share the same solution set. If without any optimization, the number of the solution space areas would be considerable huge, which might significantly increase the computational effort. So SPA also contains a group of optimization methods to reduce the amount of the solution space areas. Hence, we can get the mapping from the aggregation functions to the solution sets, which can greatly improve the query efficiency. Finally, extensive experiments show that our algorithms have significant improvement on the efficiency, compared with Threshold Algorithm, Density Threshold Algorithm and Selective-Density Threshold Algorithm.

Keywords: Algorithms, Information Retrieval, Top-k Query, Reverse Top-k Query, Solution Prediction

第一章 引言

第一节 背景综述

Top-k 检索算法研究[4][5]，是信息检索领域的一个分支研究课题。近年来，随着网络的普及和信息爆炸带来的数据库规模的急剧增长，Top-k 检索问题受到了越来越多的重视和关注。Top-k 检索问题有着非常广泛的实际应用背景，比如网页搜索引擎[6][7][10][11]、数据库检索查询[6][7][8][9]、多媒体数据库信息管理[8][9]等等。Top-k 检索算法具有较强的研究价值，除了与其广泛而强大的应用背景有关之外，一个理论上的主要原因是，Top-k 检索算法可以显著地摒弃掉与用户或者与实际检索需求无重要关联的大量无关信息与对检索结果没有任何影响的无关数据项，从而较为明显地提高信息检索与查询的质量和效率。

然而 Top-k 检索算法也有较为明显的局限与不足，这也使得它往往需要加以包装、优化与进行相应的改进之后，才能真正应用到其广泛的实际应用背景中。首先，Top-k 检索算法需要将所有数据均用数字化表示，比如各个物体表示为相应的数据项，物体的各个属性表示为数据集中的各个维度，物体的属性状态与各态标志必须表示为数字化的本地分数等等。但是很多情况下，现实生活中的一些实体与状态标志是很难表示为数字的，或者表示为数字之后会发生失真的情况导致不可忽略的误差。除此之外，Top-k 检索算法模型要求输入的检索形式为严格单调递增或严格单调递减的聚合函数，而且通常为严格单调递增的聚合权函数。然而通常情况下，首先并不是所有检索都是可以表示为数学意义上的聚合函数的，其次严格单调递增或严格单调递减的聚合函数这一要求通常也不是广泛地可以达到的。另外 Top-k 检索算法模型要求空间维数有限等，也属于一些局限性的表现。

第二节 本文的工作

离线优化处理[1][2][3][5]是 Top-k 检索算法研究领域的一个重要分支，目前的离线优化工作多是对数据库的裁剪[3]，或是建立索引以辅助实时算法的执行[1][2][23]，然而对 Top-k 检索解空间的预知性研究却很少。

本文旨在研究 Top-k 检索问题解空间的预知性问题。即通过对 Top-k 检索的解空间的相关推断，来提高实时 Top-k 检索的效率。

2011 年提出的 Reverse Top-k 检索[15]衍生于经典 Top-k 问题，可描述为：在二维情况下，即数据集的属性值的个数为二，对二维数据库中的某一项，求出聚合权函数的集合，使得在实时 Top-k 检索查询时，当聚合权函数为此集合中的一员时，此项为 Top-k 检索结果中的一解。其引入了较为直观的几何意义：二维平面中，将聚合权函数表示为二维向量 f，同时将所给定的二维数据集中的所有二维数据项表示为平面中的一点（使用点的平面坐标表示其在各属性的本地分数，必要时可以进行归一化），经过给定点 p 的垂直于聚合权函数向量 f 的直线为 p 的“优胜线”，此“优胜线”将平面分割为两个区域，左下区域中的所有数据点的 Top-k 检索排名均劣于数据项 p。Reverse Top-k 检索问题并未受到关注，但其本质是对 Top-k 检索解空间的预知性问题。

因此，Reverse Top-k 检索可以作为 Top-k 检索算法的离线优化处理，求出每个项的 Reverse Top-k 检索的解集，线上处理时，若聚合函数属于某个项的 Reverse Top-k 检索解集，则此数据项为此 Top-k 检索查询的一个解。逐点比较，算法复杂度为 O(n)。 然而 Reverse Top-k 检索很难适用于高自由度的高维情况，这也是其无法推广的重要原因之一，且其对具有 n 个数据项的数据集合需要执行 n 次 Top-k 检索算法，这个代价也是可观的。

于是本文提出基于预知解的 Top-k 检索算法 Solution Prediction Algorithm（SPA），基于非常简单的几何意义：在 m 维情况下，即给定数据集合中的属性值的个数为 m，每个数据项 x 和每个聚合权函数 f 均视为 m 维向量，则 f(x)即为数据项向量 x 与聚合权函数向量 f 的向量点乘积，或者是数据项向量 x 在聚合权函数向量 f 上的投影长度。可以证明，聚合权函数向量 f 的模长度并不影响 Top-k 检索的解集组成，故本文只考虑其方向向量，可取第一卦限的单位斜截面 E 作为研究（亦可取第一卦限单位超球面作为研究对象，并无影响）。另可证，m 维数据空间中的两个数据点 x_1 和 x_2，记 P 为经过原点、以向量 x_1x_2 为法向量的 m-1 维超平面，则对于超平面 P 上任意的聚合权函数向量 f，均有 $f(x_1) = f(x_2)$。每当聚合权函数向量从超平面 P 的一侧变化到另一侧的时候，x_1 与 x_2 的 Top-k 检索排名顺序必发生对调，故超平面 P 可视为 x_1 与 x_2 的“排序分界面”。如此，单位斜截面 E 被众多排序分界面分割为许多独立区域，对于落在同一个区域内的聚合权函数，其 Top-k 检索解集中的 n 个数据项的排序是一致的，或者说，与单位斜截面 E 相交于同一个区域内的所有聚合权函数向量，拥有相同的 Top-k 检索解集结构。

然而若不进行处理，则拥有 n 个数据项，即势为 n 的数据集，将产生共计 C_n^2 个分界面，这在 n 的数值巨大的时候是很大的数目。故进行以下处理：

1、利用 Top-k 检索中的 Domination Relationship，剔除被超过 k-1 个其他数据项 Dominate 的项。

2、若几个数据点共线，则两两之间的排序分界面相同，记为一个。

3、将单位斜截面 E 上的每一个区域视为图的结点，相邻区域之间连上一边，构成一个无向图 G。取超平面 E 边界上的一个区域、即无向图 G 的一个结点 v_0，取其中的一个聚合权函数进行 Top-k 检索，求出其 Top-k 检索的解集 R。从 v_0 开始对无向图 G 进行广度优先搜索 (Breadth First Search, BFS)，每次遍历经过排序分界面 $P(x_1, x_2)$时，考察若：

a. $x_1 \in R$ 且 $x_2 \in R$，则 R 不变，两区域的 Top-k 检索解集的组成是一致的，故可合并两区域，去除此段排序分界面，因其对 Top-k 检索解集构成无影响；

b. $x_1 \notin R$ 且 $x_2 \notin R$，则 R 不变，两区域的 Top-k 检索解集的组成是一致的，故可合并两区域，去除此段排序分界面，因其对 Top-k 检索解集构成无影响；

c. $x_1 \in R$ 且 $x_2 \notin R$，则将 x_1 从 R 中移出，将 x_2 并入 R 中，成为新的 Top-k 检索解集 R，保留此段排序分界面，因其导致 Top-k 检索解集构成发生变化；

d. $x_1 \notin R$ 且 $x_2 \in R$，则将 x_2 从 R 中移出，将 x_1 并入 R 中，成为新的 Top-k 检索解集 R，保留此段排序分界面，因其导致 Top-k 检索解集构成发生变化。

此 BFS 遍历结束后，将使得单位斜截面 E 上的区域的数目最小化，并且每个区域都与一个 Top-k 检索解集形成单射。可以证明，不同区域的 Top-k 检索解集必定是不同的。整个处理过程仅需要计算一次 Top-k 检索的解集（v_0 的解），且无维数限制。

经过以上三步之后，可以构造聚合权函数向量到 Top-k 检索解集的哈希函数，实时 Top-k 检索时，输入聚合权函数，通过哈希函数可以高效地得到对应的 Top-k 检索解集，最佳情形为 O(1)，最坏情形由所取哈希函数决定。

第三节 本文组织结构

本文的结构安排如下：

首先，在第二章中，本文将对 Top-k 检索查询问题进行简介，并作出正式的严格定义，同时给出一些实际应用的例子。随后，本章将介绍 Top-k 检索算

法研究领域的相关工作与成果，并重点介绍将用于实验性能效率对照的三个 Top-k 检索算法：Threshold Algorithm [4]，Density Threshold Algorithm [2]与 Selective-Density Threshold Algorithm [2]。

在第三章中，本文将介绍 Reverse Top-k 检索问题[15]及其引入的几何意义，随即引入 Reverse Top-k Algorithm [15] (RTA)，给出相关的检索示例进行算法分析。基于 Reverse Top-k 检索问题和 RTA，本文随即提出基于 Reverse Top-k Algorithm 的 Top-k 检索算法（Top-k algorithms Based on Reverse Top-k, TBRT）。最后就 TBRT 算法的几点不足和局限性进行分析。

在第四章中，本文将提出全新的基于预知解的高性能 Top-k 检索算法 Solution Prediction Algorithm（SPA）。并就其几何意义和相关优化细节进行必要的探讨，同时对一系列相关的定理进行证明。

随后的第五章为实验验证与展示分析章节。本章将在不同数据分布与不同数据规模的数据集合上设计一系列对比实验，通过与 Threshold Algorithm，Density Threshold Algorithm 和 Selective-Density Threshold Algorithm 的对比实验，可以看出，TBRT 算法与 SPA 算法在 Top-k 检索的效率和性能上有较为显著的提高。

最后，在第六章中，将进行全文的总结和展望。对此次毕业设计和本科毕业论文中的所有内容和工作进行总结性的概述，并对这些工作与成果如何进一步提高与改进进行展望。

第二章 Top-k 检索的定义与相关工作

第一节 Top-k 检索问题简介

假设有大量的物体或者数据项，每个物体有 m 个属性，其在每个属性上对应有一个分数，称之为该物体或数据项在此属性上的本地分数（local score）。通过输入一个聚合权函数[4]（aggregate weight function），一个物体或数据项的 m 个本地分数可以被聚合到一个总的分数。然后要选者总分数最大或最小的 k 个物体。这就是 Top-k 检索查询问题的简单描述。

可以举一个现实生活实际应用中的简单例子[12]来说明 Top-k 检索查询的具体含义。以中国科学技术大学的 GPA 排名计算为例，中国科学技术大学的每个学生就是一个物体 x_i，而每一门课或者说每门考试就是一个属性，每门考试的

得分 $s_i(x)$就是该学生在这门课上所得的 GPA。那么加权 GPA 计算的聚合权函数应该是这样的：

$$USTC-GPA(x)=\sum_{i=1}^{m} w_i \cdot s_i(x)$$

其中 w_i 是每门课程的权重（实际上应该是这些课的学分数和课时数），$s_i(x)$是该学生在这门课上的成绩，可以以 GPA 的形式，也可以以具体得分的形式。

现在需要求出 GPA 排名的前 k 个学生，就是 Top-k 检索查询问题的一个最简单的应用。

Top-*k* 查询在很多领域都有运用。比如说信息检索（information retrieval）[6][7]、网络和系统监控（network and system monitoring）[8][9]、P2P 系统和传感网络（P2P systems and sensor networks） [10][11]等。近年来，随着网络的普及和信息爆炸带来的数据库规模的急剧增长，Top-k 检索问题受到了越来越多的重视和关注。Top-k 检索算法具有较强的研究价值，除了与其广泛而强大的应用背景有关之外，一个理论上的主要原因是，Top-k 检索算法可以显著地摒弃掉与用户或者与实际检索需求无重要关联的大量无关信息与对检索结果没有任何影响的无关数据项，从而较为明显地提高信息检索与查询的质量和效率。

Top-k 检索算法的普遍模型是给定一个有限的数据集合，其规模大小为 n，即具有 n 个数据项；数据集的度为 m，即属性列的个数为 m。各数据项在各属性上具有各自的本地分数，归一化处理之后取值范围为[0, 1]。检索的输入形式为严格单调递增或者严格单调递减的 m 维聚合权函数，检索的输出结果为聚合得分最高或者最低的 k 个数据项的集合。

以上的通用 Top-k 检索算法模型具有较少的限制，为了使模型更具体化，Fagin 等人提出了一个具有更多限制的具体 Top-k 检索模型[4]，简称有序属性表 Top-k 检索算法模型：数据库有 *m* 个表（list），每个表（list）有 *n* 个数据项并根据本地分数进行降序排序。每个数据项可以被顺序访问（sorted access）和随机访问(random access)。针对这一模型，Fagin 提出了 Fagin’s Algorithm [4] (FA)作为一般的求解算法。随后 Fagin 又提出了 TA 算法 [4](Threshold Algorithm)，并证明了 TA 算法的实例最优性 [4](instance optimality)，TA 算法被认为是 Top-k 检索研究领域的经典高效算法。对于某些数据库，随机访问的代价是十分巨大甚至无法实现的。针对无随机访问的 Top-k 模型，Fagin 等人提出了 No Random Access [4](NRA)算法。精确 Top-k 检索的研究一直是一个很活跃的学术领域。

第二节 Top-k 检索算法模型定义

本文的模型定义[12]如下：假设有 n 个物体（object）和 m 个表（List），记为 $L_1, L_2, ..., L_M$，每个表代表物体的属性（attribute）。每个表有 n 个数据项，每个数据项如 $(x, s_i(x))$ 其中 x 表示物体，$s_i(x)$ 表示 x 的第 i 个本地分数（local score）。每个数据项既能做顺序访问（sorted access），也能做随机访问（random access）。每个物体通过一个聚合函数（aggregation function）得到一个总的分数。Top-k 检索查询的目标就是找到 k 个总分数最高的物体（Top-k）。

其中聚合函数必须是单调函数（*Monotonic Function*）：

定义 2.1 *Monotonic Function*.[4] 聚合函数 f 是 *Monotonic Function*，如果 $f(a_1, a_2 \dots a_m) \le f(a_1', a_2' \dots a_m')$,其中对于任意 i 均有 $a_i \le a_i'$。

在本文中，假设聚合函数是加权求和函数 $f(x) = \sum_{i=1}^{m} w_i s_i(x)$，其中 $s_i(x) \in [0, 1]$ and $\sum_{i=1}^{m} w_i = 1$ $(w_i \neq 0)$.

第三节 Threshold Algorithm

针对经典的 Top-k 检索模型，Fagin 等人相应地提出了 TA 算法[4]（Threshold Algorithm），并证明了 TA 的实例最优性[4]（instance optimality），随后 TA 成为 Top-k 检索研究领域的经典算法。

TA 算法的描述如图一所示，其中符号如表 1 所示。

表 1　符号定义

n	物体个数	k	返回的个数
m	表的数量	L_i	第 i 个数据表
f	聚合函数	$s_i(x)$	x 的第 i 个本地分数
Y	检索结果集合	$\underline{x_i}$	第 i 个表的最底分数

TA 的基本思想是这样的：

首先在预处理过程中，将所给定的数据集中的所有属性表进行排序，使得所有属性表中的数据项均按其在该属性上的本地分数降序排列（最大 Top-k 检索情况）或升序排列（最小 Top-k 检索情况）。预处理结束后，数据集的形式即满足了 Fagin 等人提出有序属性表 Top-k 检索算法模型[4]。随后在实时检索处理过程中，执行以下三步：

Threshold Algorithm (TA)

Pre-computing Phase:

For each attribute $i \in \{1, 2 \ldots m\}$, get every $s_i(x_j)$ where $j \in \{1, 2 \ldots n\}$ and insert them into a sorted list L_i. Sorted list means that objects in each list are sorted in descending order by the $s_i(x_j)$ value.

Computing Phase:

1: $Y = \{\text{dummy}_1, \ldots \text{dummy}_k\}$ with $f(\text{dummy}_i) = 0$, $\tau = 0$, $M_k = 0$.
2: **while** ($Y.size < k$ or $\tau > M_k$) **do**
3: $(x, s_i(x)) = $ Get_next_item $(L_1, \ldots, L_m)$. // get next item from one of the lists $L_1, \ldots, L_m$ in parallel.
4: $\underline{x}_i = s_i(x)$ and $\tau = f(\underline{x}_1, \underline{x}_2 \ldots \underline{x}_m)$.
5: **if** ($x \notin Y$) **then**
6: get the missing local scores of the object x and calculate the total score $f(x)$;
7: **if** ($Y.size < k$) **then**
8: insert x into Y;
9: Let t be the object with the lowest total score in Y and M_k be its total score;
10: **else**
11: **if** ($f(x) > M_k$) **then**
12: remove t from Y and insert x into Y;
13: Let t be the object with the lowest total score in Y and M_k be its total score;
14: **Return** Y.

图一 Threshold Algorithm

1. 平行地顺序访问各个属性表，每当一个物体被顺序访问，即做随机访问后求得它的总分数，如果它是目前为止见到的 k 个得分最高的物体，则记住它和它的得分；

2. 对于每一个属性表 L_i，令 $\underline{x}_i$ 为 L_i 最后一个被顺序访问所得到的分数。定义下界 τ 为 $t(\underline{x}_1, \ldots..., \underline{x}_m)$，t 为聚合函数。当且仅当至少 k 个已被访问的物体的得分不小于 τ 时，TA 终止。

3. 令 Y 为记录当前得分最高的 k 个物体的集合，则 TA 终止时返回 Y。

表 2 数据表示例

L_1	L_2	L_3
$(x_2,0.9)$	$(x_1,0.8)$	$(x_3,0.7)$
$(x_1,0.5)$	$(x_4,0.7)$	$(x_4,0.6)$
$(x_3,0.4)$	$(x_2,0.6)$	$(x_2,0.5)$
$(x_5,0.3)$	$(x_3,0.3)$	$(x_5,0.2)$
$(x_4,0.1)$	$(x_5,0.2)$	$(x_1,0.1)$

这里使用一个示例来说明 TA 的执行过程。

示例 1： 假设 $n = 5$, $m = 3$, $k = 1$, $f = \text{sum}$。 数据集如表 2 所示。首先，TA 通过顺序访问 L_1 得到 x_2。然后 TA 通过在 L_2 和 L_3 上的随机存取找到它的第一个和第二个本地分数，接下来计算总分 $f(x_2) = 2$。 因为 $\tau = 2.9$（初始化时，$s_i =$

1）。此后，TA 继续顺序访问 L_1。 在第二次顺序访问的时候，TA 找到 x_1，然后 TA 通过随机访问 L_1 和 L_2 得到它的第一个和第二个本地分数并计算总分 $f(x_1)$= 1.4。 当前结果的集合 Y 仍然包含 x_2，但 r 被更新为 2.7。$\tau > f(x_2)$，所以 TA 继续在 L_2 上做顺序访问。 一直重复这个过程直至 TA 在 L_1 上做完 2 次顺序访问和在 L_2 与 L_3 做完一次随机访问（$\tau = 2.0 \leq \lambda = f(x_2)$）。 因此，在本例中，TA 通过 4 次随机访问和 8 次随机访问找到 Top-1 结果。

第四节 Density Threshold Algorithm 与 Selective-DTA

Density Threshold Algorithm [2]（DTA，2011 年）与其同系优化算法 Selective Density Threshold Algorithm [2]（S-DTA，2011 年）是近年提出的基于 Threshold Algorithm [4]（TA，2001 年）进行实时性能优化的高效 Top-k 检索查询算法。除了与 Top-k 检索查询的经典 TA 算法做实验对照之外，本文亦选择了 DTA 与 S-DTA 这两个年轻的高效算法作为实验对照对象，用以测试 TBRT 算法与 SPA 算法在实际应用中的效能提升情况。

本节简单地介绍 DTA 算法与 S-DTA 算法。

首先，DTA 算法的算法描述如图二所示：

Density Threshold Algorithm (DTA)

Pre-computing Phase:
Build the Density Index of the given database.
Computing Phase:
1: $Y = \emptyset$, $\tau = 0$, $M_k = 0$, $\underline{s_i} = 1$ where $i = 1, 2 \ldots m$.
2: **while** ($Y.size < k$ or $\tau > M_k$) **do**
3: Let Sn_j be the section with the largest lean value in the available sections according to Density Index.
4: **for** item $(x, s_i(x))$ = **from** the head item **down to** the tail item of Sn_j **do**
5: $\underline{s_i} = s_i(x)$.
6: $\tau = f(\underline{s_1}, \underline{s_2} \ldots \underline{s_m})$.
7: **if** object x has not been accessed before **then**
8: get the missing local scores of the object x and calculate the total score $f(x)$;
9: **if** ($Y.size < k$) **then**
10: insert x into Y;
11: Let t be the object with the lowest total score in Y and M_k be its total score;
12: **else**
13: **if** ($f(x) > M_k$) **then**
14: remove t from Y and insert x into Y;
15: Let t be the object with the lowest total score in Y and M_k be its total score;
16: **if** ($\tau \leq M_k$) **then**
17: go to 20.
18: **end for.**
19: **end while.**
20: **Return** Y.

图二 Density Threshold Algorithm

DTA 算法[2]是基于 TA 算法[4]的改进算法，首先在预处理过程中，DTA 算法执行与 TA 算法一致的预处理，将所给定的数据集中的所有属性表进行排序，使得所有属性表中的数据项均按其在该属性上的本地分数降序排列（最大 Top-k 检索情况）或升序排列（最小 Top-k 检索情况）。随后 DTA 算法在离线情况下在数据集合上建立其辅助数据结构 Density Index。

线上处理时，DTA 算法基于线下处理时构造的 Density Index [2]，每次进行类似 TA 算法中的平行访问时，DTA 算法选择是的 Threshold Value 下降最快的属性列进行访问，实现实时选择的类平行访问。

S-DTA 算法[2]是对 DTA 算法在实例最优性上的改进和优化，可以证明，DTA 算法本身不是实例最优的[2]，这会使得在某些情况下，DTA 算法会得到最坏的结果——花费最长的时间。S-DTA 算法解决了 DTA 算法中的实例最优性问题，其算法描述如图三所示：

Selective -Density Threshold Algorithm (S-DTA)

Pre-computing Phase:
Build the Density Index of the given database.
Computing Phase:
1: $Y = \emptyset$, $\tau = 0$, $M_k = 0$, *select* = 0, $\underline{s_i} = 1$ where $i = 1, 2 \ldots m$.
2: **while** ($Y.size < k$ or $\tau > M_k$) **do**
3: … // the same as line 3 in DTA.
4: **for** item $(x, s_i(x))$ = **from** the head item **down** **to** the tail item of Sn_j **do**
5: *select* = *select* + 1.
6: **if** *select* mod $v == 0$ **then**
7: **for** each $l \in \{1, 2 \ldots m\}$ **do in parallel**
8: $(x, s_l(x))$ = Get_next_item $(L_1, \ldots, L_m)$. // get next item from one of the lists L_i in parallel.
9: $\underline{s_l} = s_l(x)$.
10: … // the same as DTA from line 6 to line 17.
11: **end for.**
12: **else**
13: … // the same as DTA from line 5 to line 17.
14: **end for.**
15: **end while.**
16: **Return** *Y*.

图三 Selective Density Threshold Algorithm

可以证明，S-DTA 算法是实例最优的。由于 DTA 算法和 S-DTA 算法仅是本文中用于实验对照的两个算法，并无太大的探讨意义，故在本文中不再详细描述。

第三章 Reverse Top-k 检索、RTA 算法与 TBRT 算法

第一节 Reverse Top-k 检索问题

一、Reverse Top-k 检索问题的研究背景

如前所述，Top-k 检索查询对于输入的检索聚合权函数，返回聚合总分最高的 k 个数据项。从商业实际应用背景的角度来看，检索的聚合权函数即为用户的需求，而数据库中的数据项即为商业产品，则 Top-k 检索查询所返回的是最符合用户输入要求的 k 个产品，这是从用户、消费者角度出发的一种查询检索。然而，从商人或者企业经营者的角度，则需要与 Top-k 检索相对的一种“逆”Top-k 检索查询。这就是 Reverse Top-k query [15]（2011 年提出）的大背景。商人或者企业经营者需要考虑和追求的是，在相同的市场环境下（即相同的数据库分布状态下），使得自身的产品尽量地满足最多用户的检索要求。数学化的描述是，给定一个数据库，指定数据库中的某一项，求出聚合权函数的集合，使得在实时 Top-k 检索查询时，当聚合权函数为此集合中的一员时，此项为 Top-k 检索结果中的一解。

二、Reverse Top-k 检索问题的定义

本小段给出 Reverse Top-k 检索问题的严格定义如下：

定义 3.1 *Reverse Top-k query*. [15] 给定一个数据集，正整数 k，指定数据集中的某一项 p，*Reverse Top-k query* 返回函数集合 R’ = $RTOP_k(p)$：对于$\forall f \in R'$，均有 $p \in TOP_k(f)$，i. e.，$\exists q \in TOP_k(f)$，有 $f(q) \le f(p)$。

Reverse Top-k 检索查询的简单数学化表示为：

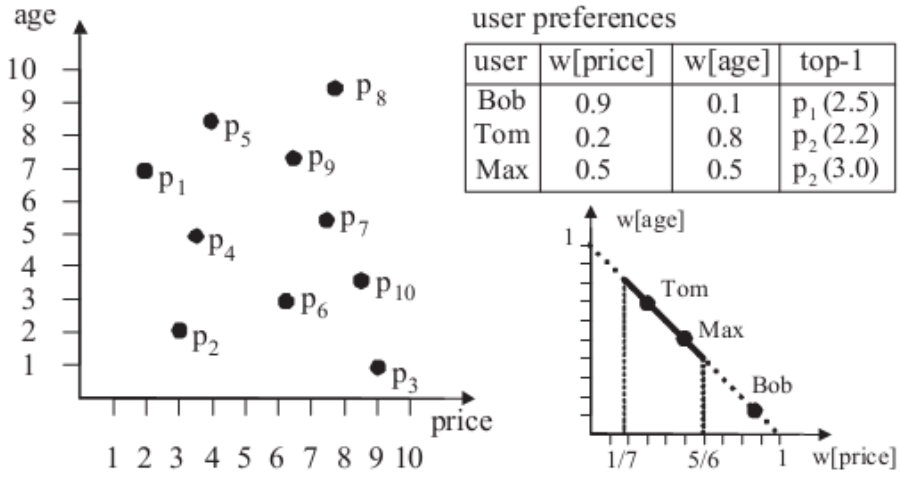
$$RTOP_k(p) = \{R'[i] \mid \forall f \in R'[i], p \in TOP_k(f)\}。$$

为了更直观地说明 Reverse Top-k 检索问题，这里举一个简单的例子来进行说明。

示例 2：如图四所示，将给定的二维数据集映射到二维平面坐标系中，每个属性表示为平面的一个维度。User preference 代表用户喜好，即用户用于进行 Top-k 实时检索查询时将输入的聚合权函数的各个权值，这里取 k 为 1，即 Top-k 检索将返回得分最高的物体或数据项。这个示例的实际应用背景是 Bob、Tom 和 Max 三人考虑买一辆二手的小汽车，这里将问题简化了，小汽车具有两个属性：价格 price 和年龄 age。三个用户根据自己的喜好给这两个属性赋以了各自

的权重，由于每人只买一辆小汽车，所以这是一个 Top-1 检索查询。

对于聚合权函数为(0.9, 0.1)的 Bob 而言，p_1 是其 Top-1 检索的结果。而 p_2 则既是 Tom 的 Top-1 检索结果，又是 Max 的 Top-1 检索结果。事实上，所有 w[price]在区间[1/7, 5/6]范围内的检索聚合权函数都使得 p_2 是 Top-1 检索的结果，也即是 p_2 的 Reverse Top-k 检索的解集。



图四　Reverse Top-k 检索查询示例

图四中的右下图即为 Reverse Top-1 检索的解空间简图，第一象限的单位截线段 w[price] + w[age] = 1 代表了所有聚合权函数的投影范围，实线部分即为 p_2 的 Reverse Top-k 检索的解集。因此，Reverse Top-k 检索的解空间是由 w[price]与 w[age]所定义的空间。

第二节 Reverse Top-k Algorithm

本章的上一节已经详细的介绍了 Reverse Top-k 检索查询问题，在这一节中，将提出在属性值个数为二的情况下，即二维数据集合的条件下，解决 Reverse Top-k 检索查询问题的算法：Reverse Top-k Algorithm [15]（RTA）。首先，第一小段将对 Reverse Top-k 检索问题在二维平面空间下的几何意义进行探讨，由此引出 RTA 的基本算法思想，最后在第二小段中对 RTA 进行详细的算法描述，并给出简单的示例用以说明算法的执行过程。

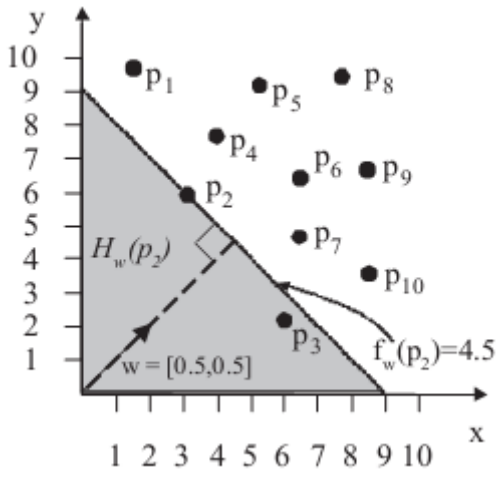
一、Reverse Top-k 检索问题的几何意义探讨

在欧几里德空间中，一个线性的 Top-k 检索查询可以被表示为一个向量 w，这个向量为聚合权函数向量，向量 w 的每一维的坐标即为聚合权函数在相

应维度的权值。

定义 3.2 *优胜线*. [15] 给定二维数据集，将其映射到二维平面坐标系中，对于某数据点 p 与聚合权函数 w，垂直于聚合权函数向量 w，且经过数据点 p 的直线，即为数据点 p 对于聚合权函数 w 的*优胜线*。

优胜线的示意图如图五，其实 w 为(0.5, 0.5)，p 点取为 p_2：



图五 优胜线示例

可以看出，优胜线将平面坐标区域经过 p 点分割为左下区域和右上区域。

定义 3.3 *优胜区域*. 已知数据点 p 对于聚合权函数 w 的优胜线，其相应的*优胜区域*为使得其中所有数据点在聚合权函数为 w 的 Top-k 检索查询中的排序名次均优于数据点 p 的区域，记为 $H_w(p)$。

定理 3.1 对于最小 Top-k 检索查询，数据点对于聚合权函数的优胜区域为左下方区域；对于最大 Top-k 检索查询，数据点对于聚合权函数的优胜区域为右上方区域。

证明：记 l 为数据点 p 对于聚合权函数 w 的优胜线。数据点 x 对于聚合权函数 w 的聚合分数为

$$f(x) = w_1 x_1 + w_2 x_2$$
$$= \vec{w} \bullet \vec{x} = \left|\vec{w}\right| \times \left|\vec{x}\right| \times \cos < \vec{w}, \vec{x} >$$

其中，记 $\vec{w} = (w_1, w_2)$，$\vec{x} = (x_1, x_2)$。因给定的聚合权函数向量 w 的模长度对于所有数据点均一致，故仅 $\left|\vec{x}\right| \times \cos < \vec{w}, \vec{x} >$ 对 Top-k 检索查询的排序结果产生影响，此项数值即为数据项向量 x 在聚合权函数向量 w 方向上的投影长度。因数据点 p 对于聚合权函数向量 w 的优胜线垂直于 w 且通过 p，可知位于优胜线左下方区域的向量在聚合权函数向量 w 方向上的投影长度均小于数据点

p 在 w 方向上的投影长度，而优胜线的右上方区域的向量在聚合权函数向量 w 方向上的投影长度均大于数据点 p 在 w 方向上的投影长度。由此可知，落于优胜线左下方区域的数据点，在聚合权函数为 w 的情况下，聚合得分均小于数据点 p，而落于优胜线右上方区域的数据点，在聚合权函数为 w 的情况下，聚合得分均大于数据点 p。

因此，对于最小 Top-k 检索查询，数据点对于聚合权函数的优胜区域为左下方区域；对于最大 Top-k 检索查询，数据点对于聚合权函数的优胜区域为右上方区域。**证毕。**

示例 3 以图五为例，聚合权函数向量 w = [0.5, 0.5]，取定点 p_2，则所有落在数据点 p_2 对于聚合权函数 w 的优胜线 l 上的数据点 p_i，其聚合得分均为 $f_w(p_i) = f_w(p_2) = 4.5$。另外，$p_2$ 亦是此数据集合上的最小 Top-2 检索 0.5x+0.5y 的结果之一，它在此最小 Top-k 检索查询中排序为第二。从图中可见，其优胜区域 $H_w(p_2)$中，仅有一个数据点 p_3。然而，若为最大 Top-k 检索查询，则数据点 p_2 对于聚合权函数 w 的优胜区域则为优胜线 l 的右上方区域，可见其最大 Top-k 检索排序为第九。

二、RTA 算法的描述及分析

本小段将介绍用于在属性值个数为二的情况下解决 Reverse Top-k 检索问题的高效算法 Reverse Top-k Algorithm [15]，简称 RTA。

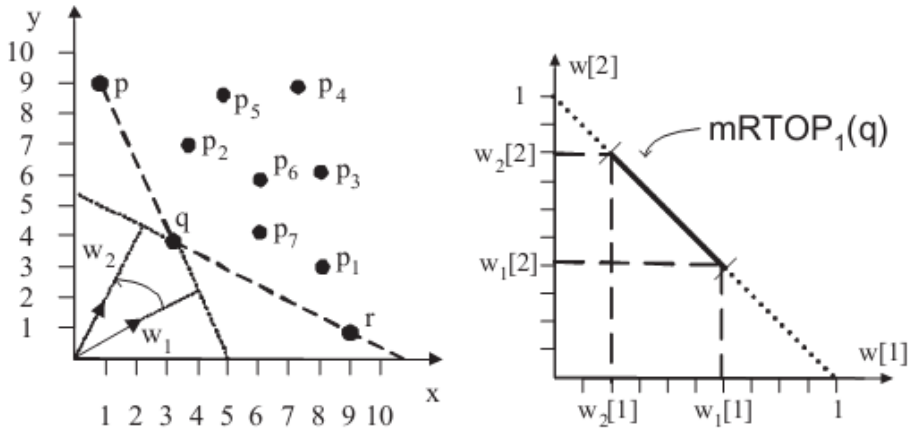
如前所述，对于给定数据点 q，其 Reverse Top-k 检索问题的解集可以表示为

$$RTOP_k(q) = \{W_i : \exists w_j \in W_i \wedge q \in TOP_k(w_j)\}$$

示例 4 考虑图六(a)中的二维数据集合的情况下，对于所有的聚合权函数，仅有 p、q、r 三点有可能成为最小 Top-1 检索的结果。故存在至少一个聚合权函数 w_i，使得 q 成为其 Top-1 检索查询的解，即 $q \in TOP_1(w_i)$。所以至少存在一个聚合权函数区间 $W_i \in RTOP_1(q)$。为了求出 W_i 的各个边界值，必须考察线段 pq 和 qr 的情况。

令 w_1 为垂直于线段 pq 的聚合权函数向量，则有 $f_{w1}(p) = f_{w2}(q)$，故在 $TOP_1(w_1)$检索中，数据点 p 和 q 有完全相同的排序。又直线 $l_w(p)$垂直于聚合权函数向量 w 且经过 p，定义了聚合得分的总分值和数据点对于 w 的 Top-1 检索的排序。对于比聚合权函数向量 w_1 具有更大或者更小的旋转角度的聚合权函数，p 和 q 的相对排序位置对调。如果对于比聚合权函数向量 w_1 具有更小的旋

转角度的聚合权函数，数据点 p 的排序低于数据点 q，则对于比聚合权函数向量 w_1 具有更大的旋转角度的聚合权函数，数据点 p 的排序高于数据点 q。因为数据点 p 和数据点 q 的相对排序位置仅对调一次，则必存在惟一一个分区间 W_i，使得对于任意聚合权函数 $w \in W_i$，均有 $q \in TOP_1(w)$成立。



(a) Data Space　　(b) Solution Space.

图六　RTA 算法思路推导示例

此唯一分量 W_i 是由聚合权函数向量 w_1 和 w_2 决定的，其中 w_1 为垂直于线段 pq 的聚合权函数向量，w_2 为垂直于线段 qr 的聚合权函数向量。此数据集合中，数据点 q 的 Reverse Top-1 检索的解集中，所有的聚合权函数 w 均满足以下不等式：

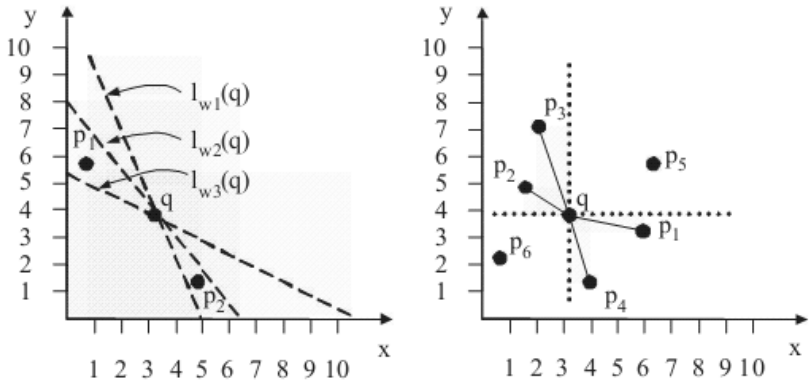
$$\frac{\lambda_{qr}}{\lambda_{qr}-1} \le w[1] \le \frac{\lambda_{pq}}{\lambda_{pq}-1}$$

其中，$\lambda_{pq} = \frac{q[2]-p[2]}{q[1]-p[1]}$ 与 $\lambda_{qr} = \frac{r[2]-q[2]}{r[1]-q[1]}$ 分别是直线 pq 和直线 qr 的斜率。以上的不等式是由 $w_1 \perp pq$ 与 $w_2 \perp qr$ 推断而来的。在图六(b)中，数据点 q 的 Reverse Top-1 检索的解集为二维平面空间坐标系中第一象限单位斜截线 w[1] + w[2] = 1 上标出的实线部分。

由此，本段给出 Reverse Top-k Algorithm（RTA）的算法描述如图八所示：

在此，本段给出一个 RTA 算法的执行示例，便于详细说明 RTA 算法的执行过程：

示例 5 对于图七(b)中的示例，对数据点 q 进行 Reverse Top-k 检索，执行 RTA 算法过程如下：p_5 的 Top-k 检索排序必定劣于 q，而 p_6 的 Top-k 检索排序必定优于 q，这对于任意的聚合权函数 w 都是成立的，下一章中将解释这一规律为 Domination Relationship。所以第五行中实际上是将所有 Dominate 数据点 q



(a) Multiple partitions. (b) $mRTOP_k(q)$ algorithm.

图七 RTA 算法执行示例

Algorithm 1 Monochromatic Reverse top-k Algorithm.

1: **Input:** S, q
2: **Output:** $mRTOP_k(q)$
3: $W' \leftarrow \{\emptyset\}$, $R \leftarrow \{\emptyset\}$, $RES \leftarrow \{\emptyset\}$
4: **for** $(\forall p_i \in S)$ **do**
5: **if** $(q \not\prec p_i$ **and** $p_i \not\prec q)$ **then**
6: $w_i[1] \leftarrow \frac{\lambda_{p_i q}}{\lambda_{p_i q}-1}$, , $w_i[2] \leftarrow 1 - w_i[1]$
7: $W' \leftarrow W' \cup \{w_i\}$
8: **end if**
9: **end for**
10: sort W' based on $w_i[1]$
11: $w_0 \leftarrow [0, 1]$, $w_{|W'|+1} \leftarrow [1, 0]$
12: $R \leftarrow \{p : p$ *lies in* $H_{w_0}(q)\}$
13: $k_w \leftarrow |R|$ //number of points in R
14: **for** $(\forall w_i \in W')$ **do**
15: **if** $(k_w \leq k)$ **then**
16: $RES \leftarrow RES \cup \{(w_i, w_{i+1})\}$
17: **end if**
18: **if** $(p_{i+1} \in R)$ **then**
19: $k_w \leftarrow k_w - 1$
20: **else**
21: $k_w \leftarrow k_w + 1$
22: **end if**
23: **end for**
24: **return** RES

图八 RTA 算法描述

或者被数据点 q 所 Dominate 的数据点从考虑范围内剔除，因为它们对于数据点 q 的 Top-k 检索排序的相对位置不会改变，不具比较意义。幸运的是，使得数据点 p_i 与数据点 q 的 Top-k 检索排序位置对调的聚合权函数 w_i，可以由直线 qp_i 的垂线所决定。所以，RTA 算法对所有直线 qp_i 进行考察，其中 p_i 是与数据点 q

不具有任何 Domination Relationship 的所有数据点。这些直线决定了解集 W 的各个分量 W_i 的各个边界值，故在第七行中，将所有的这些聚合权函数 w_i 保存在列表 W'中。因此，RTA 通过对列表 W'进行处理来求解 q 的 Reverse Top-k 检索的各个解。

在图七(b)中，经过第十行的排序之后，可以得到列表 W'为{w_1, w_2, w_3, w_4}，其排列顺序分别对应于直线 qp_1, qp_2, qp_3, qp_4。随后，第十一行中，w_0 和 w_5 被加入到列表 W'中。在第十二行中，对于第一个聚合权函数向量 w_0，所有在其优胜区域 $H_{w0}(q)$中的数据点都被访问。又如前所证，落在优胜区域 $H_{w0}(q)$内的数据点的个数决定了数据点 q 在基于聚合权函数 w_0 的 Top-k 检索中的排序（第十三行）。在这个例子中，Top-k 检索的初始解集 R 是{p_4, p_6, p_1}，故数据点 q 的初始 Top-k 检索排序为 4。在经过聚合权函数向量 w_1 之前，q 的 Top-k 检索排序的位置不会改变。这里，可以设 k = 3，则对于第一个聚合权函数分量 $W_0 = [w_0, w_1]$，数据点 q 的 Top-k 检索排序高于 k，所以分量 W_0 可以被剔除。随后，在下一个分量 $W_1 = [w_1, w_2]$中，因为数据点 $p_1 \in R$（第十八行），这意味着在区间 W_1 中，数据点 p_1 和 q 的相对 Top-k 检索排序位置对调，故数据点 q 的 Top-k 检索排序为 3。在第十六行中，W_1 被加入到解集 $RTOP_3(q)$中。依此类推，RTA 算法考察了所有聚合权函数分量 W_i 中数据点 q 的 Top-k 检索排序。在本例中，W_1 即为 $RTOP_3(q)$中唯一一个聚合权函数分量。

第三节 Top-k algorithm Based on Reverse Top-k

在本节中，将介绍基于 Reverse Top-k 检索查询的 Top-k 检索算法：Top-k algorithm Based on Reverse Top-k，简称 TBRT 算法，如图九所示。

如前所述，Reverse Top-k 检索问题并未受到关注，但其本质是对 Top-k 检索解空间的预知性问题。这符合本文的研究目的——通过对 Top-k 检索的解空间的预知性研究来提高实时 Top-k 检索的效率与性能。因此，Reverse Top-k 检索可以作为 Top-k 检索算法的离线优化处理，求出每个项的 Reverse Top-k 检索的解集，线上处理时，若聚合函数属于某个项的 Reverse Top-k 检索解集，则此数据项为此 Top-k 检索查询的一个解。逐点比较，算法复杂度为 O(n)。于是，本节提出基于 Reverse Top-k Algorithm 的 Top-k 检索算法：

Reverse Top-k Algorithm 的正确性保证了 TBRT 算法的正确性。

Top-k algorithm Based on Reverse Top-k (TBRT)

Pre-computing Phase:

For each item p in the given dataset, run RTA to get $RTOP_k(p)$.

Computing Phase:

1: $R = \emptyset$, note the aggregation weight function as f.
2: **for each** item p **do**
3: **if** $f \in RTOP_k(p)$ **then**
4: $R \leftarrow R \cup \{p\}$.
5: **if** $R.size \geq k$ **then**
6: **Return** R.
7: **end if.**
8: **end if.**
9: **end for.**

图九 TBRT 算法描述

第四章 基于预知解的高性能 Top-k 算法

第一节 几何意义探讨与相关定理

一、 最底层的几何意义：向量点乘积

在第三章中，探讨了 Reverse Top-k 检索问题中，在二维情况下的几个重要的几何意义：优胜线与优胜区域，并相应地证明了一个定理。事实上，Reverse Top-k 检索中的优胜线与优胜区域的几何意义虽然直观，但却不够简洁，这也是使得 Reverse Top-k 检索查询问题算法：Reverse Top-k Algorithm（RTA）的应用范围被限制与二维情况下的一个重要原因。当属性值的个数为多个的时候，Reverse Top-k 检索的解空间处于高自由度的高维情形下，这使得 RTA 算法执行的主要思想变得十分复杂，难以实现。

为了使得算法即基于对解空间的预知性研究问题，又可以推广应用于高维情况而不受维数的限制，必须重新设定整个算法所依赖的几何意义。为了摆脱高维数带来的高自由所导致的各种限制，算法所依赖的几何意义必须尽量地简单化、尽量地底层化，使得附加的限制最小化。

对于聚合权函数为 f 的 m 维 Top-k 检索查询，将给定的 m 维数据集合映射到对应的 m 维空间中，每一 m 维数据项 x 与空间中的一个 m 维数据点 p 对应，点 p 的每一维的坐标值均为数据项 x 在相应属性上的本地分数。将聚合权函数 f 表示为聚合权函数向量 w，其中向量 w 的起点为坐标系原点，指向以聚合权函数 f 各维所赋权值为各维度坐标值的点。同时将所有数据点 p 表示为数据向量

p，其中向量 p 的起点为坐标系原点，指向数据点 p。故有：

引理 4.1 在以上欧几里德几何坐标系设定下，对于聚合权函数为 f 的 m 维 Top-k 检索查询，数据项 x 的聚合总得分 f(x)为向量 w 与向量 p 的点乘积，即 $f(x)=\vec{w}\bullet\vec{p}$。

证明：易证得

$f(x)=\sum_{i=1}^{m} f_i\cdot x_i=\sum_{i=1}^{m} w_i\cdot p_i=\vec{w}\bullet\vec{p}$，故得证。**证毕。**

引理 4.2 聚合权函数向量 w 的模长度，不影响 Top-k 检索查询的解集构成。

证明： 对于数据项 x 与数据项 y，其中 x 在 m 维空间中的映射数据点为 p，y 在 m 维空间中的映射数据点为 q。由于聚合权函数向量 w 的模长度对于所有数据项都是一致的，故有：

$$f(x)\le f(y)\Leftrightarrow\sum_{i=1}^{m} f_i\cdot x_i\le\sum_{i=1}^{m} f_i\cdot y_i$$

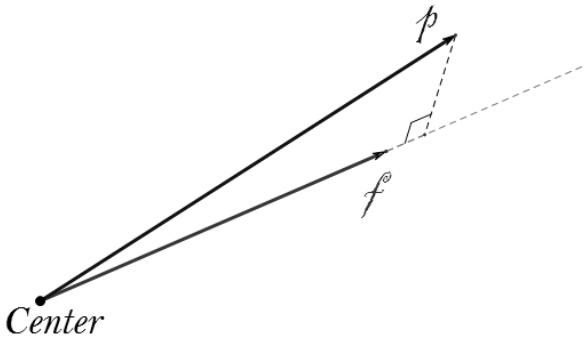
$$\Leftrightarrow\sum_{i=1}^{m} w_i\cdot p_i\le\sum_{i=1}^{m} w_i\cdot q_i$$

$$\Leftrightarrow\vec{w}\cdot\vec{p}\le\vec{w}\cdot\vec{q}$$

$$\Leftrightarrow\left|\vec{w}\right|\times\left|\vec{p}\right|\times\cos<\vec{w},\vec{p}>\le\left|\vec{w}\right|\times\left|\vec{q}\right|\times\cos<\vec{w},\vec{q}>$$

$$\Leftrightarrow\left|\vec{p}\right|\times\cos<\vec{w},\vec{p}>\le\left|\vec{q}\right|\times\cos<\vec{w},\vec{q}>$$

由上可见，两数据项 x 与 y 的聚合总得分的大小关系与聚合权函数向量 w 的模长度无关，仅决定于对应的数据向量 p 与 q 在聚合权函数向量 w 的方向向量上的投影长度（如图十所示）。**证毕。**



图十　数据向量在聚合权函数向量上的投影

将聚合权函数和数据项的聚合运算，转换为两个向量的点乘积，是非常简单与底层的几何意义，不受任何维数的自由度限制，是一个可行的方案。然

而，这仅是最底层的几何意义，不足以构成算法的支撑，还需要在此基础上定义更多的几何意义。

二、 排序分界面的引入

定义 4.1 *排序分界面.* 在 m 维欧几里德空间中，考察两个数据点 p 与 q。令 P 为经过坐标系原点、以向量 pq 为法向量的 m – 1 维超平面，则超平面 P 为数据点 p 与 q 的*排序分界面*。

“排序分界面”的几何意义顾名思义，是两个数据点 p 与 q 在 Top-k 检索查询中的的排序位置的分界面，即若对于终点落在排序分界面 P 一侧的聚合权函数向量 w，有 $f_w(p) \le f_w(q)$， 则对于终点落在排序分界面 P 另一侧的聚合权函数 v，必有 $f_v(p) \ge f_v(q)$。下面将证明此段推论的正确性。

定理 4.3 在 m 维情况下，令 m-1 为超平面 P 为 m 维数据点 p 与 q 的排序分界面，则对于任意终点位于超平面 P 上的聚合权函数向量 w，均有 $f_w(p) = f_w(q)$ 成立。

证明：记 m 维数据点 p 的坐标值为$(p_1, p_2, \ldots, p_m)$，q 的坐标值为$(q_1, q_2, \ldots, q_m)$，则 m-1 维超平面 P 以向量 $pq = (q_1 - p_1, q_2 - p_2, \ldots, q_m - p_m)$为法向量，且排序分界面 P 经过坐标系原点$(0, 0, \ldots, 0)$。故可得排序分界面 P 的点法式方程为：

$$P = \{x \mid (q_1 - p_1)x_1 + (q_2 - p_2)x_2 + \ldots + (p_m - q_m)x_m = 0\}$$

由此可知，对于终点位于超平面 P 上、起点为坐标系原点的聚合权函数向量，有

$$\begin{aligned}\overrightarrow{W_P} &= \{\vec{w} \mid (q_1 - p_1)w_1 + (q_2 - p_2)w_2 + \ldots + (q_m - p_m)w_m = 0\} \\ &= \{\vec{w} \mid \sum_{i=1}^{m}(q_i - p_i)w_i = 0\} \\ &= \{\vec{w} \mid \sum_{i=1}^{m} q_i w_i = \sum_{i=1}^{m} p_i w_i\} \\ &= \{\vec{w} \mid f_{\vec{w}}(\vec{q}) = f_{\vec{w}}(\vec{p})\}\end{aligned}$$

故对于任意终点位于超平面 P 上的聚合权函数向量 w，均有 $f_w(p) = f_w(q)$成立。**证毕。**

定理 4.4 在 m 维情况下，令 m – 1 维超平面 P 为两 m 维数据点 p 与 q 的排序分界面，则在超平面 P 的两侧，数据点 p 与 q 的 Top-k 检索排序的相对位置相反。即若对于终点落在排序分界面 P 一侧的聚合权函数向量 w，有 $f_w(p) \le f_w(q)$， 则对于终点落在排序分界面 P 另一侧的聚合权函数 v，必有 $f_v(p) \ge$

$f_v(q)$。

证明： 根据定理 4.3，对于终点落在排序分界面 P 上的任意聚合权函数向量 w，数据点 p 与 q 的聚合得分均相等，即有 $f_w(p) = f_w(q)$成立。

取超平面 P 的一侧 P_{Left}，其空间方程为：

$$P_{Left} = \{x \mid (q_1 - p_1)x_1 + (q_2 - p_2)x_2 + \ldots + (p_m - q_m)x_m \le 0\}$$

故对于终点落在空间区域 P_{Left} 中的任意聚合权函数向量 w 的集合 W_{PLeft}，有

$$\begin{aligned}\overrightarrow{W_{PLeft}} &= \{\vec{w} \mid (q_1 - p_1)w_1 + (q_2 - p_2)w_2 + \ldots + (q_m - p_m)w_m \le 0\} \\ &= \{\vec{w} \mid \sum_{i=1}^{m}(q_i - p_i)w_i \le 0\} \\ &= \{\vec{w} \mid \sum_{i=1}^{m} q_i w_i \le \sum_{i=1}^{m} p_i w_i\} \\ &= \{\vec{w} \mid f_{\vec{w}}(\vec{q}) \le f_{\vec{w}}(\vec{p})\}\end{aligned}$$

即对于集合 W_{PLeft} 中的任意聚合权函数向量 w，均有 $f_w(p) \le f_w(q)$成立，即数据点 p 的 Top-k 检索排序不优于 q（最大 Top-k 检索情况）。

同理，对于超平面 P 的另一侧 P_{Right}，其空间方程为：

$$P_{Right} = \{x \mid (q_1 - p_1)x_1 + (q_2 - p_2)x_2 + \ldots + (p_m - q_m)x_m \ge 0\}$$

故对于终点落在空间区域 P_{Right} 中的任意聚合权函数向量 v 的集合 V_{PRight}，有

$$\begin{aligned}\overrightarrow{V_{PRight}} &= \{\vec{v} \mid (q_1 - p_1)v_1 + (q_2 - p_2)v_2 + \ldots + (q_m - p_m)v_m \ge 0\} \\ &= \{\vec{v} \mid \sum_{i=1}^{m}(q_i - p_i)v_i \ge 0\} \\ &= \{\vec{v} \mid \sum_{i=1}^{m} q_i v_i \ge \sum_{i=1}^{m} p_i v_i\} \\ &= \{\vec{v} \mid f_{\vec{v}}(\vec{q}) \ge f_{\vec{v}}(\vec{p})\}\end{aligned}$$

即对于集合 W_{PLeft} 中的任意聚合权函数向量 w，均有 $f_w(p) \ge f_w(q)$成立，即数据点 p 的 Top-k 检索排序不劣于 q（最大 Top-k 检索情况）。

故在排序分界面 P 的两侧，数据点 p 与 q 的 Top-k 检索排序的相对位置相反。**证毕。**

排序分界面的另一种理解是，根据引理 4.2，数据点 p 与 q 的 Top-k 检索排序仅与其相应数据向量在聚合权函数向量方向上的投影的长度有关，而排序分界面 P 经过坐标系原点、垂直于向量 pq，则导致当聚合权函数向量 w 位于超平面 P 的一侧时，数据点 p 在聚合权函数向量 w 方向上的投影大于数据点 q 的投

影，而当聚合权函数向量 w 位于超平面 P 的另一侧时，数据点 p 在聚合权函数向量 w 方向上的投影小于数据点 q 的投影。在聚合权函数向量 w 上的投影长度的变化直接导致了数据点 p 和 q 的 Top-k 检索排序的变化。

排序分界面的定义及其性质将几何意义与 Top-k 检索的排序相关联，这对 Top-k 检索的解空间的预知性研究问题很有帮助。下面引入的更多的几何意义与定理，可以更进一步地对 Top-k 检索的解空间进行预测。

三、 聚合权函数空间的保序区域分解

如引理 4.2 所述，聚合权函数向量 w 的模长度，不影响 Top-k 检索查询的解集构成。故本文只考察聚合权函数向量的方向向量，则只需在空间中选定一个特殊的超平面，使得所有聚合权函数向量的可能方向向量都穿过此超平面，故此超平面覆盖了聚合权函数向量的所有取值，从而构成了聚合权函数的函数空间。所以本文仅需研究此超平面的状况，即可考察到整个聚合权函数的函数空间。

上述设定的超平面可以有很多选择，比如 m 维空间中圆心为坐标系原点的单位超球面的第一卦限部分，即 $1/2^m$ 单位超球面。为了简洁方便，不带来太多的计算复杂性，本文中选定第一卦限的单位斜截面 E，有

$$E = \{x \mid x_1 + x_2 + \ldots + x_m = 1 \text{ 且 } x_i \geq 0\}$$

易知单位斜截面 E 满足设定要求，是聚合权函数的函数空间的一个双射。结合本节第二小段引入的排序分界面，可知单位斜截面 E、即聚合权函数的函数空间被众多排序分界面分割为许多封闭区域，记为$\{A_i\}$。三维情形下的示意图如图十一所示。

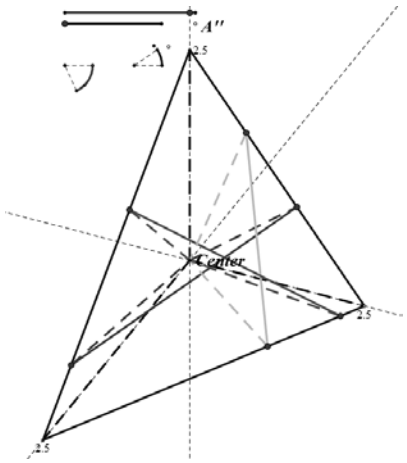
下面将研究聚合权函数的函数空间被排序分界面所分割出来的区域集合$\{A_i\}$的性质和特点。

定义 4.2 *保序区域.* 以上设定中，单位斜截面 E、即聚合权函数的函数空间被排序分界面所分割出来的各个区域 A_i 称为*保序区域*。

下面两个定理说明了保序区域的重要性质。

定理 4.5 对于方向向量的终点落在同一个保序区域内的所有聚合权函数向量 w，数据集合中的所有数据项的 Top-k 检索排序均一致。

证明：（反证法）考察某保序区域，对于该保序区域内的两个聚合权函数向量 w 与 v，满足：向量 w 与向量 v 不平行，即其方向向量不是平行向量，且向量 w 与向量 v 的方向向量均穿过同一保序区域。



图十一 三维情形下的保序区域示意图

若存在数据项 p，其在基于聚合权函数 w 的 Top-k 检索中的排序与其在基于聚合权函数 v 的 Top-k 检索中的排序不一致，此处假设在最大 Top-k 检索中，数据项 p 在基于聚合权函数 w 的 Top-k 检索中的排序优于其在基于聚合权函数 v 的 Top-k 检索中的排序。

因数据集合不变，数据项的总数 n 不会发生变化，则必存在至少一个不同于 p 的数据项 q，满足：$f_w(p) \geq f_w(q)$且 $f_v(p) \leq f_v(q)$。则经过本节第二小段对排序分界面的讨论及定理 4.4，聚合权函数向量 w 与 v 必位于数据点 p 与 q 的排序分界面的两侧。这说明此排序分界面穿过此保序区域，这与保序区域的定义 4.2 相矛盾，故不存在这种情况。

其他所有情形同理可证。**证毕。**

定理 4.5 说明了保序区域的一个重要性质：保序性，对于方向向量的终点落在同一个保序区域内的所有聚合权函数向量 w，数据集合中的所有数据项的 Top-k 检索排序均一致。

引理 4.6 对于数据点 p 与数据点 q 的排序分界面 P，当聚合权函数向量 w 从排序分界面 P 的一侧微元变化到另一侧时，其产生的 Top-k 检索排序中 p 与 q 的位置发生对调，且 p 与 q 的排序位置必定是相邻的。即不存在另一数据点 r，使得 $f_w(q) < f_w(r) < f_w(p)$或 $f_w(q) > f_w(r) > f_w(p)$。

证明： 根据定理 4.4，当聚合权函数向量 w 从排序分界面 P 的一侧微元变化到另一侧时，其产生的 Top-k 检索排序中 p 与 q 的位置发生对调。若存在另一数据点 r，使得 $f_w(q) < f_w(r) < f_w(p)$或 $f_w(q) > f_w(r) > f_w(p)$。则在聚合权函数向量 w 从 p 与 q 的排序分界面 P 的一侧微元变化到另一侧时，$f_w(q)$与 $f_w(p)$的大小

关系发生对调，由于聚合权函数 f_w 为连续函数，$f_w(q)$与 $f_w(p)$将互相逼近直至相等，随后互相分开。故数据点 q 与 r、数据点 p 与 r 的 Top-k 检索位置也将发生对调，这与聚合权函数向量 w 从排序分界面 P 的一侧微元变化到另一侧的条件矛盾。故假设不成立，**证毕。**

下面的定理将说明保序区域与 Top-k 检索排序空间的特殊对应关系。

定理 4.7 保序区域的集合到 Top-k 检索的排序组合空间的映射是一个单射。

证明：（反证法）假设：保序区域的集合到 Top-k 检索的排序组合空间的映射不是一个单射。则存在不同的保序区域 A 与保序区域 B，方向向量的终点落在保序区域 A 或保序区域 B 中的聚合权函数所产生的 Top-k 检索的排序是一致的。

不妨取保序区域 A 中的聚合权函数 w，与保序区域 B 中聚合权函数 v，若假设成立，则有聚合权函数为 f_w 时的 Top-k 检索排序与聚合权函数为 f_v 时的 Top-k 检索排序完全一致。分类讨论：

若保序区域 A 与 B 相邻，则根据保序区域的定义 4.2，保序区域 A 与 B 的邻边必是一个排序分界面，这说明保序区域 A 与保序区域 B 分别位于一个排序分界面的两侧，即聚合权函数向量 w 与聚合权函数向量 v 分别位于一个排序分界面的两侧。则根据定理 4.4，必存在两个数据点 p 与 q，使得对于聚合权函数向量 w 与聚合权函数向量 v，数据点 p 与 q 的 Top-k 检索排序的相对位置相反。与假设的推论矛盾，故假设不成立。

若保序区域 A 与保序区域 B 不相邻，则对于从区域 A 通往区域 B 的任一路径，必定穿过任一排序分界面偶数次，才能保证所有数据项的 Top-k 检索排序保持不变。而根据引理 4.6，此路劲必定为一个封闭环路，即保序区域 B 即为保序区域 A 本身，两者为同一个保序区域，这与假设矛盾，故假设不成立。

因此，保序区域的集合到 Top-k 检索的排序组合空间的映射是一个单射。**证毕。**

向量点乘积、排序分界面和保序区域是本算法中非常重要的几何意义，向量点乘积构成了算法最底层的几何基础，而排序分界面将几何意义与 Top-k 检索排序联系起来，是算法思想的重要桥梁。保序区域则是对聚合权函数的函数空间的分解，是算法最重要的几何意义。

第二节 优化处理：从保序区域到保解区域

一、优化处理的必要性

在本章的第一节中，介绍了三个对算法非常重要的几何意义：向量点乘积、排序分界面和保序区域。保序区域直接与算法的执行相关，体现了对 Top-k 检索解空间预知性。然而，若不经过任何优化处理，将导致的直接困难就是保序区域包含过多的无用信息，且数目庞大，不方便处理。

事实上，根据本章第一节的探讨、定理 4.5 与定理 4.7，可以看出保序区域有着非常严格的性质与特点：对于方向向量的终点落在同一个保序区域内的所有聚合权函数向量 w，数据集合中的所有数据项的 Top-k 检索排序均一致。并且，保序区域的集合到 Top-k 检索的排序组合空间的映射是一个单射。

这些严格的性质，对于实时 Top-k 检索查询而言，并非必要。很多时候，虽然大部分数据项的 Top-k 检索排序发生了很大的变化，对 Top-k 检索解集的构成依然没有影响，即 Top-k 检索解集的构成保持不变。而保序区域的集合到 Top-k 检索的排序组合空间的单射性质亦非必要，这对 Top-k 检索的解集构成也没有必要的影响。

因此，对本章第一节中提出的初始几何意义进行优化处理是必要的，甚至是重要的。

首先，本节提出保解区域，这个比保序区域更为宽松的几何意义。

定义 4.3 *保解区域.* 保持区域内的所有聚合权函数向量产生的 Top-k 检索的解集的一致性的区域称为*保解区域*。

可见，对于方向向量的终点落在同一个保解区域内的所有聚合权函数向量 w，产生的所有 Top-k 检索查询具有相同的解集构成。

至此，本节提出三点优化处理的方案，优化的目的是减少数据量、剔除不必要的排序分界面与最小化保解区域的数目。

二、 Dominate Degree 优化处理

当数据量庞大的时候，并非所有的数据项都有可能成为 Top-k 检索的一个解的，对于很多数据项，存在至少 k 个不同的数据项，在任何聚合权函数下的聚合得分均高于它们，则这些数据项可以被剔除，不被考虑，用以大量减少数据量。由于数据项数目的大量减少，排序分界面的数目也会大大减少，从而保序区域的数量也会骤减。

首先，此处定义 Domination Relationship 如下（在最大 Top-k 检索查询情况下）：

定义 4.4 *Dominate* [7]. 数据项 x dominate 数据项 y，当且仅当以下条件同时成立：

（1） 对于任意 i, $i \in \{1, 2, \ldots, m\}$，均有 $x_i \geq y_i$；

（2） 至少存在一个 j, $j \in \{1, 2, \ldots, m\}$，使得 $x_j \geq y_j$。

基于 Dominate 的定义，可以提出 Domination Relationship 的衍生定义：Dominate Degree 如下：

定义 4.5 *Dominate Degree* [13]. 如果一个数据项 x 被 i 个数据项 dominate，则它的 Dominate Degree 为 i，记为 dd(x)=i。

下面提出两个定理，这两个定理很好地说明了 Dominate 和 Dominate Degree 在 Top-k 检索查询中的重要性质。

推论 4.8 若数据项 x 被另一数据项 y 所 dominate，则对于任意非零聚合权函数 f，均有 f(y) > f(x)。

证明： 根据定义 2.1，聚合函数 *f* 是 *Monotonic Function*，如果 $f(a_1, a_2 \ldots a_m) \leq f(a_1', a_2' \ldots a_m')$,其中对于任意 *i* 均有 $a_i \leq a_i'$。由根据定义 4.4，可得 f(y) ≥ f(x)。又 f 为非零聚合权函数，则有 f(y) > f(x)。**证毕。**

推论 4.9 若数据项 x 的 dominate degree 大于 k – 1，则对于任意聚合权函数，x 均不可能成为 Top-k 检索的一个解。

证明：由定义 4.5 可知，若数据项 x 的 dominate degree 大于 k – 1，则存在至少 k 个不同的数据项 dominate 数据项 x，故根据推论 4.8，对任意聚合权函数，均存在至少 k 个数据项的 Top-k 检索排序高于数据项 x，数据项 x 均无法成为 Top-k 检索的一个解。**证毕。**

由推论 4.9 可知，在进行预处理时，可以事先将所有 dominate degree 大于 k – 1 的数据项均剔除，不做考虑，方可大量减少需要处理的数据量，从而减少各数据项所产生的排序分界面的数量以及保序区域的数目，降低处理难度。

三、 多点共线的情况

根据排序分界面的定义 4.1，排序分界面为经过坐标系原点、以两点所组成的向量为法向量的 m – 1 维超平面，当存在多于两个点共线的情况时，这些点均具有相同的排序分界面，可以将它们合并为一个。

四、 合并保序区域到保解区域

如前所述，保序区域有着非常严格的性质与特点：对于方向向量的终点落在同一个保序区域内的所有聚合权函数向量 w，数据集合中的所有数据项的 Top-k 检索排序均一致。并且，保序区域的集合到 Top-k 检索的排序组合空间的映射是一个单射。

这些严格的性质，对于实时 Top-k 检索查询而言，并非必要。很多时候，虽然大部分数据项的 Top-k 检索排序发生了很大的变化，对 Top-k 检索解集的构成依然没有影响，即 Top-k 检索解集的构成保持不变。而保序区域的集合到 Top-k 检索的排序组合空间的单射性质亦非必要，这对 Top-k 检索的解集构成也没有必要的影响。

因此，本小段作出的优化处理即为合并保序区域到保解区域。

具体的处理过程如下：

将单位斜截面 E 上的每一个保序区域视为图的结点，相邻保序区域之间连上一边，构成一个无向图 G。取超平面 E 边界上的一个保序区域、即无向图 G 的一个结点 v_0，取其中的一个聚合权函数进行 Top-k 检索，求出其 Top-k 检索的解集 R。从 v_0 开始对无向图 G 进行广度优先搜索 (Breadth First Search，BFS)，每次遍历经过排序分界面 $P(x_1, x_2)$时，考察若：

a. $x_1 \in R$ 且 $x_2 \in R$，则 R 不变，合并两保序区域，去除此段排序分界面；

b. $x_1 \notin R$ 且 $x_2 \notin R$，则 R 不变，合并两保序区域，去除此段排序分界面；

c. $x_1 \in R$ 且 $x_2 \notin R$，则将 x_1 从 R 中移出，将 x_2 并入 R 中，保留此段排序分界面；

d. $x_1 \notin R$ 且 $x_2 \in R$，则将 x_2 从 R 中移出，将 x_1 并入 R 中，保留此段排序分界面。

经过以上过程的 BFS 之后，各区域的组成将发生较大的变化，区域的数目随着不断的合并而大量减少。

下面，证明这个处理过程的确是合并保序区域到保解区域，否则，这一处理将对保持实时 Top-k 检索的正确性造成影响。

定理 4.10 经过以上处理，所得的区域均为保解区域。

证明： 首先，BFS 开始与选定的结点 v_0，它代表了一个未经处理的、原始的保序区域。处理过程第一步，计算出结点 v_0 这一原始保序区域的 Top-k 检索的解集。由于保序区域是保解区域的充分条件，故 v_0 本身也是一个保解区域。

随后，在 BFS 的过程中，每次遍历经过排序分界面 $P(x_1, x_2)$时，根据定理 4.4，x_1 与 x_2 的 Top-k 检索排序发生对调，则分类讨论：

a. $x_1 \in R$ 且 $x_2 \in R$，则 x_1 与 x_2 的 Top-k 检索排序发生对调对 Top-k 检索的解集 R 的构成没有影响，故 R 不变，两保序区域的 Top-k 检索解集的组成是一致的，故可合并两保序区域，去除此段排序分界面，因其对 Top-k 检索解集构成无影响，此区域仍为保解区域；

b. $x_1 \notin R$ 且 $x_2 \notin R$，则 x_1 与 x_2 的 Top-k 检索排序发生对调对 Top-k 检索的解集 R 的构成没有影响，故 R 不变，两保序区域的 Top-k 检索解集的组成是一致的，故可合并两保序区域，去除此段排序分界面，因其对 Top-k 检索解集构成无影响，此区域仍为保解区域；

c. $x_1 \in R$ 且 $x_2 \notin R$，则 x_1 与 x_2 的 Top-k 检索排序发生对调将对 Top-k 检索的解集 R 的构成造成影响，故将 x_1 从 R 中移出，将 x_2 并入 R 中，成为新的 Top-k 检索解集 R，保留此段排序分界面，因其导致 Top-k 检索解集构成发生变化，此段排序分界面的两侧均为保解区域；

d. $x_1 \notin R$ 且 $x_2 \in R$，则 x_1 与 x_2 的 Top-k 检索排序发生对调将对 Top-k 检索的解集 R 的构成造成影响，故将 x_2 从 R 中移出，将 x_1 并入 R 中，成为新的 Top-k 检索解集 R，保留此段排序分界面，因其导致 Top-k 检索解集构成发生变化，此段排序分界面的两侧均为保解区域。

综上所述，经过以上 BFS 之后，所得的区域均为保解区域。此过程合并保序区域到保解区域。**证毕。**

因此，以上的 BFS 过程保持了保解区域保解性的传递性。

本算法中优化处理过程的目的，是合并保序区域到保解区域中，并且使得保解区域的数目最小化。下面证明，经过以上三步优化处理之后，保解区域的数目达到了最小化。

定理 4.11 上述优化处理过程执行完毕之后，保解区域的数目达到了最小化。

证明： （反证法）假设经过上述优化处理之后，保解区域的数目依然可以通过某种算法或可运算函数达到更少，则说明仍存在可以合并的保解区域。

设保解区域 A 与保解区域 B，分类讨论：

若保解区域 A 与保解区域 B 相邻，根据保序区域的定义 4.2，且保解区域是有保序区域合并而成的，则保解区域 A 与保解区域 B 之间的所有相邻面均为

排序分界面。考察其中任一排序分界面 P，设超平面 P 为数据点 x_1 与 x_2 的排序分界面，根据上述 BFS 优化处理过程的描述，a、b 两种情况下，算法均将去除排序分界面，故排序分界面 P 只可能是 c、d 两种情况下被算法所保留下来的。分类讨论：

若是情况 c：$x_1 \in R$ 且 $x_2 \notin R$，则在排序分界面 P 的两侧，数据点 x_1 与 x_2 的 Top-k 检索排序对调，即将变化为 $x_1 \notin R$ 且 $x_2 \in R$，说明 Top-k 检索的解集 R 的组成发生了变化，在排序分界面 P 的两侧，Top-k 检索的解集 R 不一致，如果将超平面 P 两侧的区域合并，则不保持保解性，无法构成保解区域。

若是情况 d：$x_1 \notin R$ 且 $x_2 \in R$，与情况 c 的证明同理，在排序分界面 P 的两侧，Top-k 检索的解集 R 不一致，如果将超平面 P 两侧的区域合并，则不保持保解性，无法构成保解区域。

故假设不成立。

若保解区域 A 与保解区域 B 不相邻，则区域 A 或者区域 B 其一必然需要通过合并其相邻的区域并最后与区域 B 合并，而如前所证，合并相邻保解区域后无法保持保解性。故假设亦不成立。

因此，上述优化处理过程执行完毕之后，保解区域的数目达到了最小化。**证毕。**

通过以上几个定理与描述，可以看出，经过本节的优化处理之后，可以较为显著地减少检索的数据量，并减少排序分界面的个数，同时将带有过于严格的性质特点的保序区域合并为更为宽松的保解区域，从而达到大大降低运算成本和存储成本的目的。

第三节 Solution Prediction Algorithm

本章第一、第二节花费了大量的篇幅对几何意义、相关定理和优化处理进行了详细入微的探讨，为基于预知解的 Top-k 检索算法（Solution Prediction Algorithm，SPA）提供了全面的理论支持与算法的正确性证明。

在本节中，将对 SPA 算法作出正式的算法描述，作为本章的总结，SPA 算法的算法描述如图十二所示。

首先，在算法的第一步，如本章第二节中第二小段中 Dominate Degree 优化处理所述，将数据集合中所有 dominate degree 大于 k – 1 的数据项，从而减少不

必要的计算量。随后，在 SPA 算法的第二步中，对于每一个数据点对，求出它们各自的排序分界面，同时根据本章第二节第三小段中多点共线情况的优化策略，将多点共线的相同排序分界面合并为一个。在算法的第三步中，取第一卦限的单位斜截面 E，结合上一步求出的排序分界面，求出所有初始保序区域 $\{A_i\}$。此后，进行本章第二节第四小段中的优化处理方案。构造无向图 G，对其进行广度优先搜索 BFS，同时分情况将保序区域合并为保解区域。最后，可以构造聚合权函数向量到 Top-k 检索解集的哈希函数，实时 Top-k 检索时，输入聚合权函数，通过哈希函数可以高效地得到对应的 Top-k 检索解集。

Solution Prediction Algorithm (SPA)

Pre-computing Phase:

1: Get $D_k = \{p: \text{dominate degree}(p) < k \mid p \in D\}$.

2: For each point pair $\{(p_i, p_j): p_i \in D_k, p_j \in D_k\}$, get their "*Rank Watershed*" $W(p_i, p_j)$, where W is a hyper-plane crossing the centre and perpendicular to vector $p_i p_j$.

3: Let hyper-plane E be the unit oblique section of the first octant, which is partitioned into many areas by the *Watersheds*. Note these areas as $\{A_i\}$.

4: Construct graph G, whose vertices are $\{A_i\}$. Add an edge between A_i and A_j iff they are neighbours. Get Top-k answers of A_0, noted as R. Run BFS on G from A_0.

5: During BFS, when crossing $W(p_i, p_j)$, **if**

5.a: $\{p_i \in R \text{ and } p_j \in R\}$ or $\{p_i \notin R \text{ and } p_j \notin R\}$ **then**
merge the two areas and keep R unchanged.

5.b: $p_i \in R$ and $p_j \notin R$ **then**
$R \leftarrow (R - \{p_i\}) \cup \{p_j\}$.

6: Construct a proper hashing function H from the aggregation weight functions to the solution sets.

Computing Phase:

Input the aggregation function f into H and get the solution set.

图十二 Solution Prediction Algorithm 算法描述

第五章 算法高效性的实验验证

第一节 实验设计综述

本章为本文的实践部分——实验验证部分，本文的前四章均为理论层次上的探讨、分析与证明，然而算法的高效性需要实验数据作为有力的实践支持。

在本章中，将设计四大组实验，分别以数据库规模 n、数据库维度 m、检

索结果规模 k 和数据库分布形态作为控制变量，对 5 个算法：Threshold Algorithm (TA)， Density Threshold Algorithm (DTA)， Selective-Density Threshold Algorithm (S-DTA)，Top-k algorithms Based on Reverse Top-k (TBRT)与 Solution Prediction Algorithm (SPA)进行对比，用以证明本文提出的两个新算法——TBRT 算法与 SPA 算法的高效性。

第二节 实验参数设定与数据库描述

本节介绍本次实验的默认实验参数的设定以及采用的一系列数据库的描述。本次实验的所有算法的代码均使用 C/C++语言编程实现，所实现的算法均在一个 8 处理器的计算机上运行，该计算机拥有 8GB 的共享内存，每一个中央处理器均为 4 核心的 Intel Xeon E5430 处理器，主频为 2.66GHz。

本组实验采用控制变量法进行对比实验设计，分别将数据库规模 n、属性值个数 m 和检索结果规模 k 作为变量，同时生成了三个具有不同数据分布的数据库，用以进行更客观的对比观察。各个实验参数的默认值如下表所示：

表 3 实验默认参数设置

Parameters	Default Values
The number of objects, i.e. *n*	10,000
The number of lists, i.e. *m*	2
The number of results returned, i.e. *k*	50
The value of *b*	100
The value of *v*	10

同时使用斐波那契序列构造聚合权函数。记 F_i = 1, 1, 2, 3, 5 …为斐波那契数，则聚合权函数为 $f(x)=\sum_{i=1}^{m}\frac{F_i}{SUM}\bullet x_i$，其中$SUM=\sum_{i=1}^{m}F_i$。

访问代价模型的设定：除 SPA 算法之外，其他四个算法（TA、DTA、S-DTA、TBRT）均以对数据库中数据项的访问次数为访问代价。SPA 算法则以对保解区域的访问次数作为访问代价。

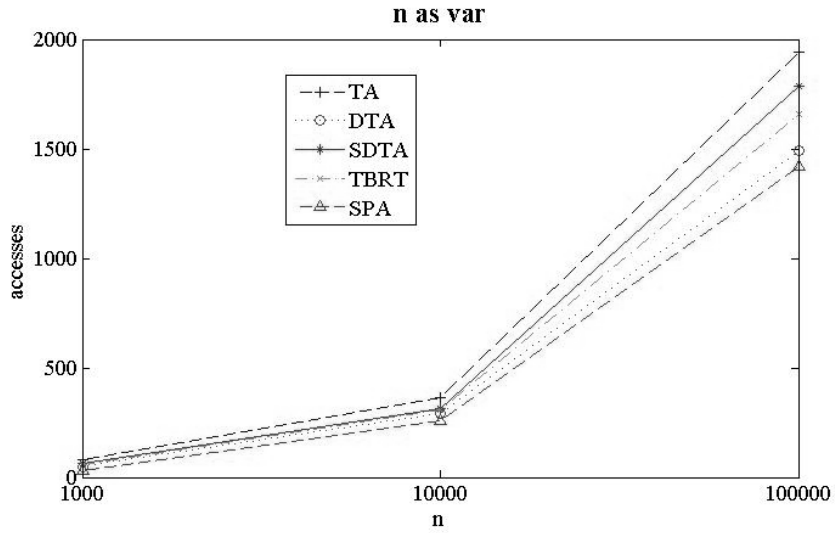
本实验共测试了三个合成的数据集 UI、NI 和 CO。其中默认数据库 UI 表示数据库中每个属性表中的本地分数呈现均匀分布（uniform distribution），不同的属性表相互独立。NI 表示每个属性表中的本地分数呈现正态分布（normal distribution），不同的表相互独立。CO 表示不同的属性表中的本地分数呈正相关的关联分布（correlated distribution），亦即当一个数据项在某个属性上具有较

高的得分时，它在其他的属性上也相应地有较高的得分，这与学生的各个学科的成绩分布模型类似。

第三节 实验结果与分析

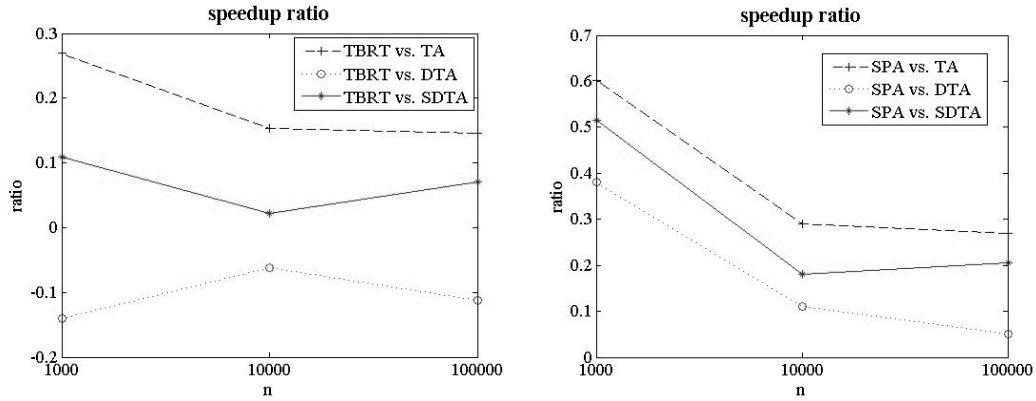
一、 数据库规模 n 为变量的对比实验

在第一套实验中，令数据库规模，即数据项的个数 n 为变量，其他实验参数均为表三中的默认参数，在 UI 数据库上执行三组对比实验，数据库规模 n 分别为 1,000、10,000 和 100,000 时进行对照比较。实验结果如图十三所示：



图十三　数据库规模 n 为变量的对比实验

TBRT 算法与 SPA 算法对于 TA 算法、DTA 算法和 S-DTA 算法的效能提升比率图分别如图十四所示。



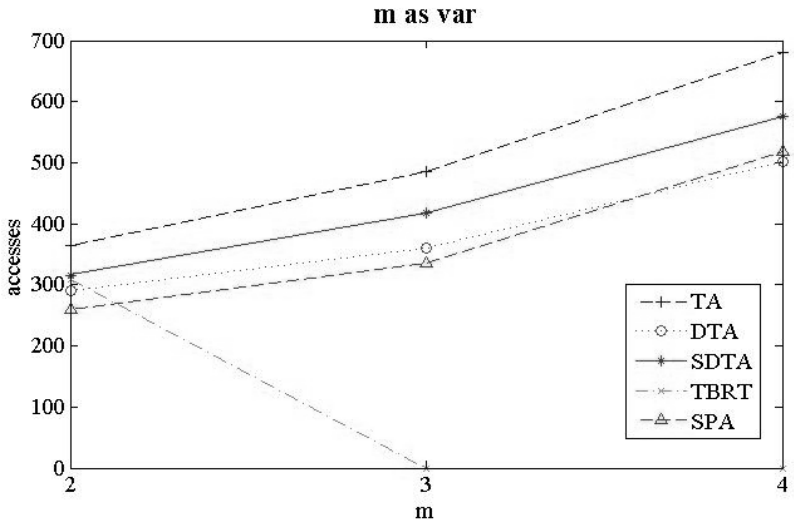
图十四　TBRT / SPA VS. TA / DTA / S-DTA

由实验结果可以得知，TBRT 算法的性能始终显著优于 TA 算法、略优于 S-

DTA 算法，并劣于 DTA 算法。SPA 算法则始终优于其他的所有算法，但在 n 值巨大的时候对于 DTA 算法的性能优势变小。

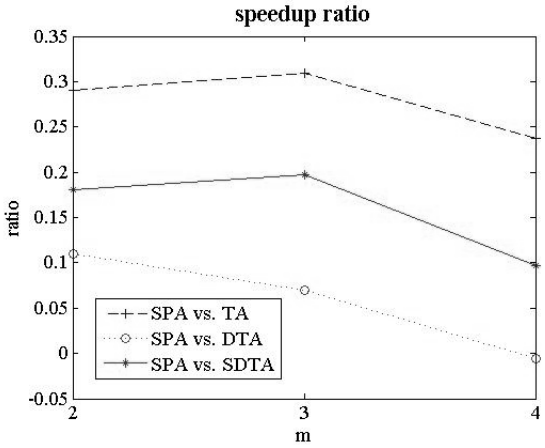
二、 属性个数 m 为变量的对比实验

在第二套实验中，令数据库维度，即属性表的个数 m 为变量，其他实验参数均为表三中的默认参数，在 UI 数据库上执行三组对比实验，数据库维度 m 分别为 2、3 和 4 时进行对照比较。实验结果如图十五所示。



图十五 属性个数 m 为变量的对比实验

SPA 算法对于 TA 算法、DTA 算法和 S-DTA 算法的效能提升比率图如图十六所示。

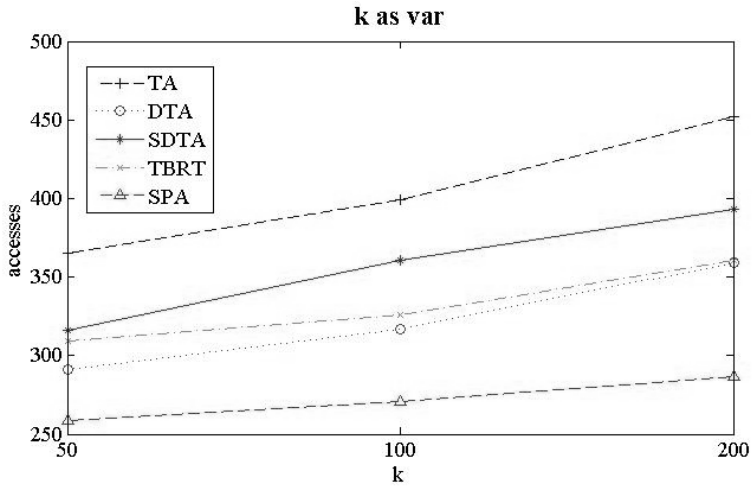


图十六 SPA VS. TA / DTA / S-DTA

由于 TBRT 算法受 RTA 算法的限制，无法应用于高维情况，故在高维数据库中并未将 TBRT 算法作为性能比较的一个成员。从实验结果数据可以看出，SPA 算法总体上仍优于所有其他算法，但在 m=4 的高维情况下，SPA 算法略劣于 DTA 算法约 0.6%。

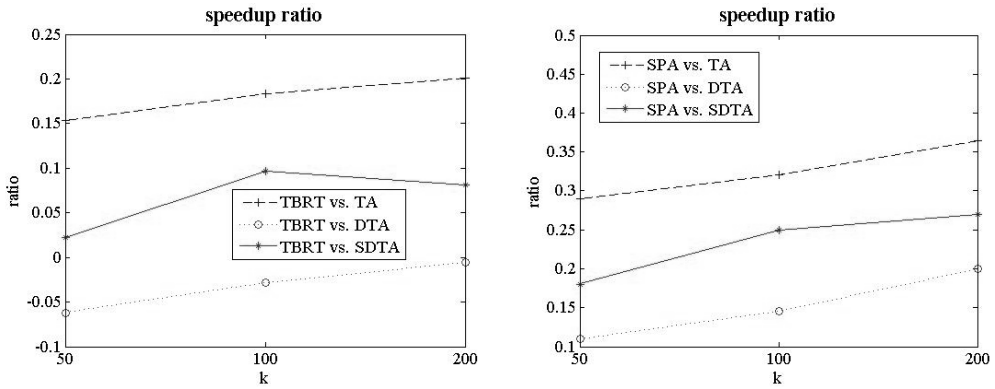
三、 检索结果规模 k 为变量的对比实验

在第三套实验中，令检索结果规模 k 为变量，其他实验参数均为表三中的默认参数，在 UI 数据库上执行三组对比实验，检索结果规模 k 分别为 50、100 和 200 时进行对照比较。实验结果如图十七所示：



图十七　检索结果规模 k 为变量的对比实验

TBRT 算法与 SPA 算法对于 TA 算法、DTA 算法和 S-DTA 算法的效能提升比率图分别如图十八所示。

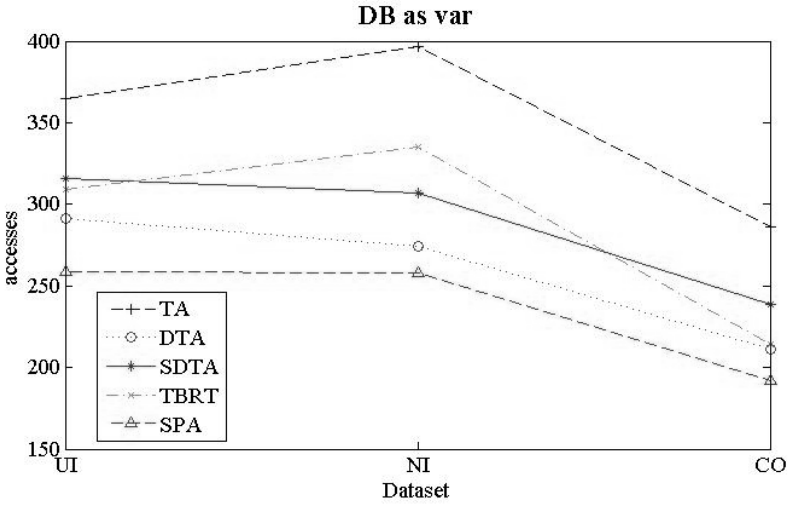


图十八　TBRT / SPA VS. TA / DTA / S-DTA

由实验结果数据可知，检索结果规模 k 对于 TA 算法、DTA 算法和 S-DTA 算法的性能影响较大，而对于 TBRT 算法与 SPA 算法的性能影响较小，如随着 k 值的增大，TBRT 算法与 SPA 算法的优越性变得明显。

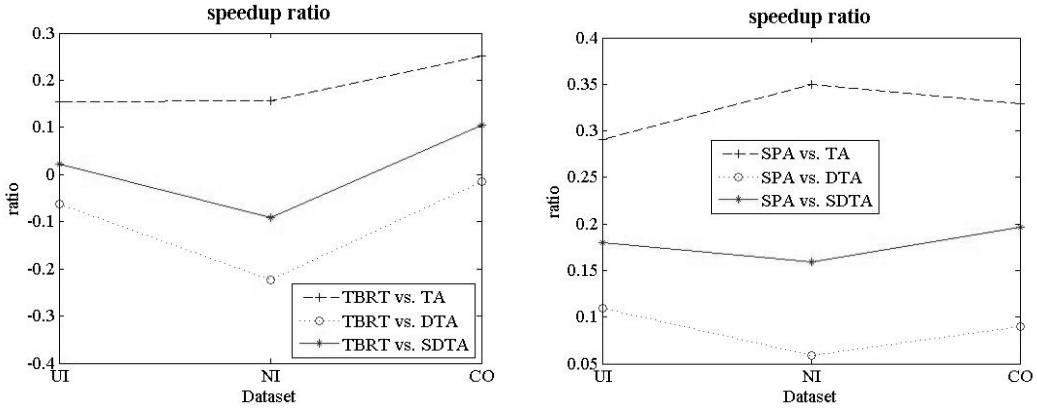
四、 在不同数据分布的数据库上的对比实验

在第四套实验中，令所有实验参数均为表三中的默认参数，在三个具有不同数据分布形态的数据库 UI、NI 和 CO 上执行三组对比实验。实验结果如图十九所示。



图十九　在不同数据分布的数据库上的对比实验

TBRT 算法与 SPA 算法对于 TA 算法、DTA 算法和 S-DTA 算法的效能提升比率图分别如下：



图二十　TBRT / SPA VS. TA / DTA / S-DTA

由实验结果图可以看出，各算法在数据呈正相关分布的数据库 CO 上表现均为最好，在加速比率上，TBRT 算法仍劣于 DTA 算法，而 SPA 算法则优于其他所有算法。TBRT 算法与 SPA 算法在数据库 CO 上均有最佳的表现。

五、 实验结果总结

本章通过设计四组对照实验，从实践层次上证明了 TBRT 算法与 SPA 算法的性能优越性。总体而言，根据以上四组实验数据的汇总结果，TBRT 算法相对于 TA 算法效率平均提升 19.4%，对于 DTA 算法效率平均提升-8.3%（劣于 DTA 算法），对于 S-DTA 算法效率平均提升 5.6%；而 SPA 算法对于 TA 算法效率平均提升 34.4%，对于 DTA 算法效率平均提升 12.1%，对于 S-DTA 算法效率平均提升 23.2%。

第六章 总结与展望

本文的主旨在于通过对 Top-k 检索解空间的预知性问题的研究，提高实时 Top-k 检索的性能与效率。本文首先对全文的核心问题：Top-k 检索查询问题进行了严格的定义与规范，随后描述了 Top-k 检索问题的相关工作：Threshold Algorithm (TA)，Density Threshold Algorithm (DTA)与 Selective Density Threshold Algorithm (S-DTA)，这也是三个本文在对比实验中用于做性能对照的三个算法。此后，本文定义了另一个检索问题：Reverse Top-k 检索问题，并介绍了解决 Reverse Top-k 检索问题的算法——Reverse Top-k Algorithm (RTA)。察觉到 Reverse Top-k 检索问题与预知解问题的关联性，本文提出了基于 RTA 算法的衍生算法：Top-k algorithms Based on Reverse Top-k (TBRT)。然而，由于 RTA 算法仅能运行在属性值个数为二的情形下，这是一个很严格的限制。为了解除这个限制，本文提出了基于预知解的高性能 Top-k 检索算法：Solution Prediction Algorithm (SPA)。为此，本文定义了一套全新的几何意义并证明了相关的定理，从理论层次上证明了 SPA 算法的正确性与高效性。最后，通过一系列的实验验证，从实践层次上证明了 TBRT 算法和 SPA 算法的高效性。

Top-k 检索问题的算法研究依然是一个无尽的课题，预知解问题的研究也是一个相对空白的方向。在以后的工作中，可以考虑将 RTA 算法进行改进，使其可以应用于高维数的情况，从而 TBRT 算法也得到了相应的改进，不再受到维数限制。另外，可以考虑将 Top-k 检索问题与 KNN 问题、Skyline 检索问题等相关问题相联系，或许可以得到更为统一、整洁的高效算法。

参 考 文 献

[1] D. Q. Chen, G. Z. Sun, Z. Q. Gong: Efficient Approximate Top-k Query Algorithm Using Cube Index, *APWeb 2011*.

[2] D. Q. Chen, G. Z. Sun, Z. Q. Gong: Efficient Top-K Query Algorithms Using Density Index, *IEEE-PACIIA 2010*.

[3] Z. Q. Gong, G. Z. Sun, D. Q. Chen: Parallel Algorithms for Top-k Query Processing, *Unpublished* .

[4] R. Fagin, A. Lotem M. Naor: Optimal Aggregation Algorithms for Middleware, *PODS*, 2001.

[5] I. Ilyas, G. Beskales, M. A. Soliman: A Survey of Top-k Query Processing Techniques in Relational Database Systems, *ACM Computing Surveys*, 2008.

[6] W.-T. Balke, W. Nejdl, W. Siberski and U. Thaden: Progressive distributed top-k retrie-val in peer-to-peer networks. In: *ICDE Conf.*, 2005.

[7] B.Kimelfeld and Y.Sagiv: Finding & Approximating Top-k Answers in Keyword Proxim-ity Search. In: *PODS Conf.*, 2006.

[8] B.Babcock and C.Olston: Distributed Top-k Monitoring. In: *SIGMOD* 2003.

[9] P.Cao and Z.Wang: Efficient Top-k Query Calculation in Distributed Networks. In: *PODC Conf.*, 2004.

[10] R.Akbarinia, E.Pacitti and P.Valduriez: Reducing Network Traffic in Unstructured P2P Systems Using Top-k Queries. In: *Distributed and Parallel Databases 19(2)*, 2006.

[11] R. Akbarinia, E. Pacitti and P. Valduriez: Processing Top-K Queries in Distributed Hash Tables. In: *Euro-Par Conf.*, 2007.

[12] D. Q. Chen: High Performance Top-k Computation Algorithm and Optimization. *USTC Undergraduate Research Program*, 2011.

[13] Z. Q. Gong, G. Z. Sun, J. Yuan,Y. Zhong: Efficient Top-*k* Query Algorithms Using *K*-skyband Partition, *INFOSCALE*, 2009.

[14] J.Yuan, G.Z.Sun, Y.Tian, G.L.Chen and Z.Liu: Selective-NRA Algorithms for Top-k Queries. In: *APWeb/ WAIM Conf.*, 2009.

[15] A. Vlachou, C. Doulkeridis, Y. Kotidis, K. Nørv˚ag: Monochromatic and Bichromatic Reverse Top-k Queries, *IEEE-TKDE*, 2011.

[16] S. Chaudhuri, N. Dalvi and R. Kaushik: Robust Cardinality and Cost Estimation for Skyline Operator, *ICDE*, 2006.

[17] I. Ilyas, R. Shah, W. Aref, J. Vitter, A. Elmagarmid: Rank-Aware Query Optimization, *ACM SIGMOD*, 2004.

[18] R. Fagin: Combining Fuzzy Information from Multiple Systems, *J. Comput. System Sci*, 58 (1), 1999.

[19] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein: *Introduction to Algorithms*. MIT Press, 2001.

[20] I. Ilyas, G. Beskales, M. A. Soliman: A Survey of Top-k Query Processing Techniques in Relational Database Systems, *ACM Computing Surveys*, 2008.

[21] D.Papadias, Y.Tao, G.Fu and B.Seeger: Progressive Skyline Computation in Database Systems. In: *ACM, Transactions on Database Systems, 30(1):41–82*, 2005.

[22] A.Vlachou, C.Doulkeridis, K.Norvag, M.Vazirgiannis: On Efficient Top-k Query Processing in Highly Distributed Environments. In: *SIGMOD Conf.*, 2008

[23] L.Zou and L.Chen: Dominant Graph: An Efficient Indexing Structure to Answer Top-K Queries. In: *ICDE Conf.*, 2008

[24] W.-T. Balke, W. Nejdl, W. Siberski and U. Thaden: Progressive Distributed Top-K Retrie-Val Ii Peer-To-Peer Networks. In: *ICDE Conf.*, 2005.

[25] D. E. Knuth: The Art of Computer Programming Volume 1: Fundamental Algorithms (Third Edition), *Addison-Wesley*, 1998.

[26] C. Buchta: On The Average Number of Maxima in a Set of Vectors. *Information Processing Letters* 33 pp.63–65, 1989.