Efficient Top-K Query Algorithms Using Density Index

Dongqu Chen[1], Guang-Zhong Sun[1,*],
Neil Zhenqiang Gong[1,**], and Xiaoqiang Zhong[2]

[1] Key Laboratory on High Performance Computing, Anhui Province School of Computer Science and Technology, University of Science and Technology of China, Hefei, Anhui, P.R. China
[2] Power Energy Metering Center, Fujian Electric Power Company Limited, Fuzhou, Fujian, P.R. China
Dong7889@126.com

Abstract. Top-*k* query has been widely studied recently in many applied fields. Fagin et al. [3] proposed an efficient algorithm, the Threshold Algorithm (i.e. TA), to process top-*k* queries. However, in many cases, TA does not terminate even if the final top-*k* results have been found for some time. Based on these, we propose a novel algorithm: Density Threshold Algorithm (i.e. DTA), which is designed to minimize the useless accesses of a top-*k* query, and introduce a novel indexing structure, *Density Index*, to support our algorithms. However, we proved the DTA is not *instance optimal* in Fagin's notion and we also propose an instance optimal algorithm named Selective-Density Threshold Algorithm (i.e. S-DTA). Finally, extensive experiments show that our algorithms have significant improvement on the efficiency, compared with the TA algorithm.

Keywords: Database query processing, Algorithms, Indexes.

1 Introduction

Ranking aware queries, or top-*k* queries, have been widely studied recently in many applied fields such as information retrieval, multimedia databases and data mining. The main reason for such attention is that top-*k* queries avoid overwhelming the user with large numbers of uninteresting answers which are resource-consuming.

A general and simple model proposed by Fagin et al. [3] is that the dataset consists of *m* sorted lists with *n* data items. Under this model, Fagin et al. [3] proposed the efficient Threshold Algorithm (i.e. TA). To measure the optimality of an algorithm, they defined a notion of optimality, *instance optimality* and proved the instance optimality of TA.

However, in many cases, TA does not terminate even if the final top-*k* results have been found for some time, which will bring more cost and delay to the queries. The

[*] Corresponding author.
[**] Neil Z. Gong is now a postgraduate in EECS, UC Berkeley. This work was completed when Neil Z. Gong was undergraduate student of USTC.

main reason for such useless accesses is the *threshold value* is not low enough to satisfy the terminate condition of TA. To speed up the reduction of the threshold value, U. Güntzer et al. [4] used the reducing speed of the accessed items to predict the speed of the unknown items in the same list. They considered that it could speed up the reduction of the threshold value if the list with the largest predicted value was chosen to be accessed in every step. Our target is similar with U. Güntzer but we reduce the threshold value through choosing the *section* with the largest *lean value* and accessing the lists *section* by *section*, based on the *Density Index* set up in the pre-computing phase.

In this paper, we study the efficient top-*k* queries using pre-computed analysis and indexing method. We propose a novel algorithm: Density Threshold Algorithm (i.e. DTA), and we also turn DTA into an instance optimal algorithm named Selective-Density Threshold Algorithm (i.e. S-DTA). Finally, extensive experiments show that our algorithms have significant improvement on the efficiency, compared with the TA algorithm.

2 Computation Model and TA Algorithm

Our model of the dataset can be described as follows [3]: assume the database D consists of m sorted lists, which are denoted as $L_1, L_2 \ldots L_m$. Each sorted list consists of n data items. We may refer to L_i as list i. Each entry of L_i is of the form $(x, s_i(x))$ where x is an object and $s_i(x)$ is the *ith* local score of x as a positive real number in the interval [0, 1]. Sorted list means that objects in each list are sorted in descending order by the $s_i(x)$ value. For a given object x, x has a total score of $f(x)=f(s_1(x), s_2(x) \ldots s_m(x))$, where the m-dimensional aggregate function f is supposed to be increasingly monotonic:

Definition 2.1 **Aggregate Monotone Function** [3]. *An aggregate function f is monotone iff $f(a_1, a_2 \ldots a_m) \le f(a_1', a_2' \ldots a_m')$, whenever $a_i \le a_i'$ for every i.*

In this paper, we assume the aggregate function is weighted summation function, yielding $f(x)=\sum_{i=1}^{m} w_i s_i(x)$ and $\sum_{i=1}^{m} w_i = 1$ ($w_i \neq 0$), the most common form of aggregate function in applications.

Each data item can be accessed through *sorted access* or *random access*. Sorted access iteratively reads data items sequentially, whereas a random access is a request for a data item in some list given the object's ID. The middleware cost of a top-*k* query algorithm is $a_S c_S + a_R c_R$, where a_S is the number of sorted accesses performed, a_R is the number of random accesses performed, c_S is the cost per sorted access, and c_R is the cost per random access.

Our task is to determine the top-*k* objects, that is, *k* objects with the highest total scores. To solve the top-*k* query described above, Fagin et al. [3] proposed the threshold algorithm (i.e. TA) as described in Fig. 1.