Expert Systems with Applications 41 (2014) 1885–1900

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Monochromatic and bichromatic mutual skyline queries

Tao Jiang [a], Yunjun Gao [b,*], Bin Zhang [a], Dan Lin [c], Qing Li [d]

[a] College of Mathematics Physics and Information Engineering, Jiaxing University, 56 Yuexiu Road (South), Jiaxing 314001, China
[b] College of Computer Science, Zhejiang University, 38 Zheda Road, Hangzhou 310027, China
[c] Department of Computer Science, Missouri University of Science and Technology, 500 West 15th Street, Rolla, MO 65409, USA
[d] Department of Computer Science, City University of Hong Kong, Tat Chee Avenue, Kowloon, Hong Kong, China

ARTICLE INFO

Keywords:
Query processing
Skyline query
Mutual skyline query
Algorithm
Spatial database

ABSTRACT

In this paper, we study a new skyline operator, namely, *mutual skyline query* (MSQ), which retrieves all the data objects that are contained in the dynamic skyline and meanwhile the reverse skyline of a specified query object q. MSQ has many applications such as marketing analysis, task allocation, and personalized matching. Motivated by this, we first formalize MSQ in both *monochromatic* and *bichromatic* cases, and then propose several algorithms for processing MSQ. Our methods utilize a conventional data-partitioning index on the dataset, employ the advantage of reusing technique, and exploit effective pruning heuristics to improve the query processing. Extensive experiments using both real and synthetic datasets demonstrate the effectiveness and efficiency of our proposed algorithms under various experimental settings.

A preliminary version of this work has been published in Zhang, Jiang, Yue, and Li (2010). Substantial new technical materials have been added to this journal submission. Specifically, the paper extends the paper (Zhang et al., 2010) by mainly including (i) an improved heap reuse algorithm, called RHBBS (Section 4); (ii) two new algorithms, called MBBS and MBBRS, which combine the BBS pass with BBRS passes into a single pass using some effective pruning strategies (Section 4); (iii) the algorithm for the bichromatic mutual skyline query (Section 5), which carries out the search on two different data sets; (iv) enhanced experimental evaluation that incorporates several new classes of queries (Section 6); and (v) a more complete review of the related work and more illustrative examples to make the paper self-contained.

Notes: (i) This manuscript is the authors' original work and has not been published nor has it been submitted simultaneously elsewhere, except for the preliminary version (i.e., Zhang et al., 2010 mentioned previously; (ii) The main differences between the preliminary version and this submission are pointed out above; and (iii) All authors have checked the manuscript and have agreed to the submission.

1. Introduction

The skyline query has attracted much attention from the database community due to its importance in a wide range of applications (Sharifzadeh & Shahabi, 2006; Balke, Güntzer, & Zheng, 2004; Hose, Lemke, & Sattler, 2006; Wu et al., 2006; Li, Tung, Jin, & Ester, 2007; Zhang, Lakshmanan, & Tung, 2009). For example, assume that a user is looking for a hotel which is near a conference venue and has cheap price. The skyline query can help the user find a list of hotels that are at least better in one of the two aspects (i.e., distance and price) compared with the remaining hotels in the database. In other words, there does not exist any hotel which is both nearer and cheaper than the other hotels in the skyline query result. In general, the skyline query returns skyline objects in a dataset that are not dominated by others in all dimensions. Besides this classic skyline query, several variants of skyline queries exist, such as the reverse skyline query (Dellis & Seeger, 2007) which retrieves objects in the database that take a given query object as their skyline objects. However, none of the existing skyline queries and variants can efficiently address the new needs (as illustrated in the following examples) emerging with the advance in social networks and cloud computing:

- **People matching**: Consider the social websites that offer dating services or friend-finding services. Users in such social websites typically have a profile describing their personal information such as age, gender, education

* Corresponding author. Address: College of Computer Science, Zhejiang University, 38 Zheda Road, Hangzhou 310027, China. Tel.: +86 571 8765 1613; fax: +86 571 8795 1250.
E-mail addresses: jxtaojiang@gmail.com (T. Jiang), gaoyj@zju.edu.cn (Y. Gao), jxbinzhang@gmail.com (B. Zhang), lindan@mst.edu (D. Lin), itqli@cityu.edu.hk (Q. Li).

0957-4174/$ - see front matter Crown Copyright © 2013 Published by Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.eswa.2013.08.085

background, and hobbies. When a user (say Bob) is looking for a date or a new friend, he may specify expectation on certain aspects. For example, Bob may want to find a friend whose age is close to him, i.e., the closer the better. He may also require that the new friend shares common interests, e.g., the degree of the interest in movie is very high. At the first look, it seems that Bob's requirement can be fulfilled by issuing a skyline query. Nevertheless, to improve the chance of Bob's invitation being accepted, it is important that the social website not only find the people who meet Bob's expectation, but also check if Bob meets those people's expectation.

- **Service matching**: Consider the two types of major players in the cloud: users and service providers. Users have a set of tasks with certain resource requirements, and they are always looking for cheaper price and better services. Cloud service providers have strength on different types of tasks and limitation on the amount of tasks, and are looking for customers who would be willing to pay more and required fewer resources. It would be beneficial for both parties if users' tasks are performed by the service providers that are users' skyline points, and meanwhile the users are also the service providers' skyline points.

To address the issue in scenarios like the above examples, a straightforward approach is to perform both a skyline query and a reverse skyline query. Consider the people matching example again. Bob can issue a skyline query to find a group of people (denoted as G1) that satisfy his expectation, and a reverse skyline query to find the group of people (dented as G2) who think Bob is a good friend candidate. Then the intersection between G1 and G2 contains the people who may also be happy to accept Bob as their new friend.

We call such a query that returns the intersection of the query results from one skyline and one reverse skyline query as *mutual skyline query* (MSQ). If the query only uses a single data set, it is referred to as *monochromatic mutual skyline query* (MMSQ). If the query involves two distinct data sets, it is called *bichromatic mutual skyline query* (BMSQ). To the best of our knowledge, this is the first time that the concept of MSQ is proposed. The mutual skyline query has a unique feature that it helps a query issuer and entities in the query result to achieve mutual benefit. We can envision that this new type of queries might bring up more interesting applications including the aforementioned two examples. Moreover, the above straightforward approach that simply applies two existing queries is very inefficient due to considerable repeated computation as discussed in Section 3.2.

In this paper, we formalize the mutual skyline query for both cases, and analyze its properties. Based on the analysis, we develop a suite of query algorithms with increasing improvement on query performance. Our methods integrate the skyline and reverse skyline query process rather than treating them separately. In this way, we can significantly reduce duplicate search and redundant computation. To this end, we propose efficient pruning heuristics and new strategies on reusing heap information. We have conducted extensive performance study on both real and synthetic datasets, and considerable experimental results have demonstrated the effectiveness and efficiency of our proposed algorithms.

The rest of the paper is organized as follows: Section 2 reviews the related work. Section 3 presents the formal definition of MSQ and presents a baseline algorithm. Section 4 elaborates our proposed query processing algorithms. Section 5 gives the bichromatic algorithms over two datasets. Section 6 reports the experimental results and our findings. Finally, Section 7 concludes the paper with some directions for future work.

2. Related work

Since the concept of the skyline query was first proposed by Börzsönyi et al. in Börzsönyi, Kossmann, and Stocker (2001), many researchers have dedicated to the study of skyline queries. In general, skyline queries can be classified into three main categories: (i) conventional static skyline query (Bartolini, Ciaccia, & Patella, 2006; Godfrey, Shipley, & Gryz, 2005; Lee, Zheng, Li, & Lee, 2007); (ii) dynamic skyline query (Deng, Zhou, & Shen, 2007); (Fuhry, Jin, & Zhang, 2009); and (iii) reverse skyline query (Dellis & Seeger, 2007; Lian & Chen, 2008). A static skyline query finds the skyline objects from a dataset P, such that every object $p \in P$ on the skyline is not dominated by any other object $r \in P \backslash \{p\}$. As an extension, the dynamic skyline query allows users to specify a query object q, and then retrieves the skyline objects with respect to q. In contrast, the reverse skyline query is to find the objects in the dataset that have q as a member of their skyline objects. In addition to three main types of skyline queries, there are some other skyline query variants (Chan, Eng, & Tan, 2005; Chan, Jagadish, Tan, Tung, & Zhang, 2006; Lin, Yuan, Wang, & Lu, 2005; Lin, Yuan, Zhang, & Zhang, 2007) such as distributed and parallel skyline queries (Gao, Chen, Chen, & Chen, 2006; Vlachou, Doulkeridis, & Kotidis, 2008; Vlachou, Doulkeridis, Kotidis, & Vazirgiannis, 2007; Vlachou, Doulkeridis, Norvag, & Vazirgiannis, 2008; Wang, Ooi, Tung, & Xu, 2007), subspace skyline queries, i.e., SKYCUBE (Yuan et al., 2005), SUBSKY (Tao, Xiao, & Pei, 2007), DADA (Li, Ooi, Tung, & Wang, 2006), SKYPEER (Vlachou et al., 2007), top-k skyline query (Lian & Chen, 2009; Vlachou et al., 2008; Yiu & Mamoulis, 2007), preference skyline query (Jiang, Pei, Lin, Cheung, & Han, 2008; Mindolin & Chomicki, 2009; Wong et al., 2008), probabilistic skyline query (Lian & Chen, 2008; Lian & Chen, 2009; Pei, Jiang, Lin, & Yuan, 2007; Zhang, Lin, Zhang, Wang, & Yu, 2009; Zhang et al., 2010), to name just a few. Since our work is more related to the three main types of skyline queries, our discussion focuses on the algorithms with respect to these three queries.

2.1. Traditional static skyline queries

Algorithms proposed for static skyline queries can be roughly classified into two categories depending on whether they use indexes or not. The first category of non-index based methods mainly includes *Divide and Conquer* (D&C) Börzsönyi et al., 2001, *Block Nested Loop* (BNL) Börzsönyi et al., 2001, *Sort Filter Skyline* (SFS) Chomicki, Godfrey, Gryz, & Liang, 2003, *Linear Elimination Sort for Skyline* (LESS) Godfrey et al., 2005, *Sort and Limit Skyline algorithm* (SaLSa) Bartolini et al., 2006, and ZSearch (Lee et al., 2007), etc. BNL (Börzsönyi et al., 2001) needs to scan the dataset a large number of passes, using a memory buffer to record the objects that are not dominated by others. D&C (Börzsönyi et al., 2001) recursively divides the dataset into partitions that fit in memory and computes the local skyline for each of them. Then, the global skyline is computed by merging all local skylines and removing dominated data points. SFS (Chomicki et al., 2003) improves BNL by pre-sorting all tuples on one of multiple dimensions with the help of a monotone function. LESS (Godfrey et al., 2005) is an optimized version of SFS and uses a small buffer to store a small set of objects in the initial pass of the external sort routine of SFS in order to prune others dominated by them early. The limitations of SFS and LESS are that they have to scan all objects at least once after sorting, which is costly. SaLSa (Bartolini et al., 2006) strives to avoid scanning the complete set of sorted objects. Unlike SFS, LESS and SaLSa, ZSearch (Lee et al., 2007) progressively processes skyline queries, in manner of Z-order search which orders the data points and clusters them in blocks to facilitate efficient dominance tests and space pruning.