Knowledge-Based Systems 89 (2015) 250–264

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

Efficient algorithms for finding the most desirable skyline objects ☆

Yunjun Gao [a,*], Qing Liu [a], Lu Chen [a], Gang Chen [a], Qing Li [b]

[a] College of Computer Science, Zhejiang University, 38 Zheda Road, Hangzhou 310027, China
[b] Department of Computer Science, City University of Hong Kong, Tat Chee Avenue, Kowloon, Hong Kong

ARTICLE INFO

Article history:
Received 29 September 2014
Received in revised form 4 April 2015
Accepted 11 July 2015
Available online 17 July 2015

Keywords:
Skyline
Query processing
Algorithm
Spatial database

ABSTRACT

The skyline query is a powerful tool for multi-criteria decision making. However, it may return too many skyline objects to offer any meaningful insight. In this paper, we introduce a new operator, namely, the *most desirable skyline object* (MDSO) *query*, to identify manageable size of *truly interesting* skyline objects. Given a multi-dimensional object set and an integer k, a MDSO query returns the *most preferable* k skyline objects, based on the *newly* defined ranking criterion that considers, for each skyline object s, the number of the objects dominated by s and their accumulated (potential) weights. We devise the ranking criterion, formalize the MDSO query, and propose three algorithms for processing MDSO queries. In addition, we extend our methods to tackle the *constrained* MDSO (CMDSO) query. Extensive experimental results on both real and synthetic datasets show that our presented ranking criterion is significant, and our proposed algorithms are efficient and scalable.

1. Introduction

Given a set P of multi-dimensional data objects, a *skyline query* returns all the data objects from P, called *skyline objects*, which are *not dominated* by any other object in P. Here, an object p *dominates* another object p' if and only if p is not worse than p' in all dimensions, and strictly better than p' in at least one dimension. The skyline query is useful in many real-life applications. Consider, for instance, a classical example of the *hotel reservation system*. Fig. 1 illustrates this case in a 2-dimensional space, where each point corresponds to a hotel record. The room *price* of a hotel is represented as the y-axis (the vertical coordinate), and the x-axis captures its *distance* to the beach (the horizontal coordinate). Hotel p_2 dominates p_5 since the former is cheaper and closer to the beach. As hotels p_1, p_2, p_3, and p_4 are not dominated by any other hotel, they constitute the *skyline* of a dataset $P = \{p_1, p_2, \ldots, p_8\}$, which offer various trade-offs between price and distance: p_4 has the cheapest room price, p_1 is the closest to the beach, and p_2, p_3 may be a good compromise of the two attributes.

Since the skyline operator was first introduced to the database community in [5], it has received considerable attention due to its wide applications related to multi-criteria decision making. A large number of algorithms have been proposed for efficient *traditional/full* skyline computation. These approaches can be mainly classified into two categories depending on whether they use indexes or not. The first one [2,5,10,17,48] involves solutions that do not assume any index on the underlying dataset, but they retrieve the skyline by scanning the entire dataset at least once. Methods of the other category [21,27,32,36] avoid accessing the whole dataset by performing the search on an appropriate index structure, e.g., an R*-tree [4]. Other variations of skyline queries include, to name just a few, *subspace* skyline computation [25,34,39], *reverse* skyline query [12,15,28], *metric* skyline computation [9,13], *continuous* skyline retrieval [19,23], *distributed* skyline computation [8,18,40], *uncertain* skyline query [33,47], skyline computation on *data streams* [29,35,38] and *incomplete data* [16], and so forth.

As pointed out in [1,7], however, the skyline query may output an *overwhelming* number of skyline objects, and thus no longer offer any interesting insights, especially in high dimensional spaces. It has been shown in [17] that, for a random dataset, the expected skyline cardinality equals $(\ln m)^{n-1}/n!$, in which m is the dataset cardinality, n represents the number of dimensions,

☆ This paper is a *full* and *extended* version of the paper, titled "*Finding the Most Desirable Skyline Objects*", which has been published (as a *short paper*) in the *proceedings of the 15th International Conference on Database Systems for Advance Applications* (*DASFAA 2010*), April 1–4, 2010, Tsukuba, Japan. Specifically, the paper extends the *short conference* paper by including (i) informative related work (Section 2); (ii) detailed the *Most Desirable Skyline Object* (MDSO) query processing algorithms, namely, CB and SB, more illustrative examples, and more analyses and proofs (Section 4); (iii) a new MDSO query processing algorithm, i.e., reused based algorithm (Section 4); (iv) processing of the *constrained MDSO* (CMDSO) *query*, a natural variant of MDSO queries (Section 5); and (v) enhanced experimental evaluation that incorporates the new type of queries (Section 6).

* Corresponding author. Tel.: +86 571 8765 1613; fax: +86 571 8795 1250.
E-mail addresses: gaoyj@zju.edu.cn (Y. Gao), liuq@zju.edu.cn (Q. Liu), luchen@zju.edu.cn (L. Chen), cg@zju.edu.cn (G. Chen), itqli@cityu.edu.hk (Q. Li).

http://dx.doi.org/10.1016/j.knosys.2015.07.007
0950-7051/

and $n!$ denotes the factorial of n. To this end, several efforts have been proposed in the literature. In particular, they control the size of skyline objects by either *relaxing the dominance relationship* or *integrating user-specific preference*, as to be surveyed in Section 2. Nonetheless, none of them takes into account the potential weights of non-skyline objects when determining the importance of skyline objects, which is certainly useful. Motivated by this, in this paper, we study *how to find the most desirable skyline objects from the skyline that consists of too many skyline objects* by considering both the number of the non-skyline objects dominated by skyline objects and the accumulated (potential) weights of non-skyline objects. Towards this, we introduce a new operator, namely, the *most desirable skyline object* (MDSO) *query*, to identify *manageable* size of *truly interesting* skyline objects.

Given a set of multi-dimensional objects and an integer k, a MDSO query returns the *most preferable* k skyline objects based on the new *ranking criterion* (defined in Definition 6) that considers, for each skyline object s, the number of the objects dominated by s and their accumulated (potential) weights. Take Fig. 1 as an example. The most desirable 1 skyline object is p_3, because it dominates the maximum number of object/points. The MDSO query is particularly helpful in web-based recommender systems. For instance, consider a tourist looking for a suitable hotel via a *web-based hotel booking system* (e.g., http://hotels.com). Most hotels in a certain city may have to be included in the skyline since, for each hotel p, there might be no one hotel that dominates p on all attributes, even if it is better than p on many attributes. On the other hand, it is difficult for the tourist to make a good, quick selection, by referencing the skyline which contains numerous hotels. In this case, MDSO queries can be employed to retrieve a few preferable hotels, such that the tourist can find a desired hotel as soon as possible.

As to be discussed in Section 2, the MDSO operator is different from the existing works. Hence, the existing techniques are not directly applicable to tackle the MDSO query efficiently. In this paper, we develop three efficient algorithms, i.e., *Cell Based algorithm* (CB), *Sweep Based algorithm* (SB), and *Reuse Based algorithm* (RB), to obtain the most desirable k skyline objects. Our methods are based on a conventional data-partitioning index (e.g., R*-tree [4]) and do not require any preprocessing. Consider that, in some real-life applications, users might enforce some constraints (e.g., spatial region) on MDSO queries. Thus, we extend our techniques to handle a natural variant of MDSO queries, namely, the *constrained most desirable skyline object* (CMDSO) *query*, which returns the *most preferable* k skyline objects in a specified *constrained region*. We also present three efficient algorithms, viz., *Constrained Cell Based algorithm* (CCB), *Constrained Sweep Based algorithm* (CSB), and *Constrained Reuse Based algorithm* (CRB), to deal with CMDSO retrieval.

In brief, the key contributions of this paper are summarized as follows:

- We devise a new ranking criterion and formalize the MDSO query, a new addition to the family of skyline operators for the skyline size control.
- We propose three algorithms, i.e., CB, SB, and RB, for efficiently processing MDSO queries, and analyze their correctness and complexities, respectively.
- We investigate a MDSO query variation, namely, CMDSO retrieval, and present three efficient algorithms to tackle it.
- We conduct extensive experiments with both real and synthetic datasets to demonstrate the effectiveness of our devised ranking criterion and the performance of our proposed algorithms in terms of efficiency and scalability.

The rest of the paper is organized as follows. Section 2 reviews related work. Section 3 formalizes our studied problem. Section 4 elaborates the three algorithms for answering MDSO queries using an R*-tree index on the dataset, and analyzes their correctness and complexities, respectively. The CMDSO query and its processing algorithms are described in Section 5. Extensive experimental evaluation and our findings are reported in Section 6. Finally, Section 7 concludes the paper with some directions for future work.

2. Related work

In this section, we briefly survey the previous work on the skyline query and its variants, and then the skyline operators for skyline size control.

The skyline operator, also known as the maximal vector problem [22], was first introduced into the database community in [5]. Since then a number of algorithms for conventional (i.e., full) skyline queries have been proposed in the literature. They can be mainly classified into (i) non-index based approaches and (ii) index based methods, depending on whether they use indexes or not. Non-index based solutions, including Block-Nested-Loop (BNL) [5], Divide and Conquer (D&C) [5], Sort-First-Skyline (SFS) [10], Linear Elimination Sort for Skyline (LESS) [17], Sort and Limit Skyline algorithm (SaLSa) [2], and Object-based Space Partitioning (OSP) [48], do not require any index on the data set to compute skyline. Index based techniques, including Bitmap [36], Index [36], Nearest Neighbor (NN) [21], Branch and Bound (BBS) [32], and ZSearch [27], require specific indexes for skyline retrieval. It has been proved [32] that BBS is I/O optimal, i.e., it accesses fewer disk pages than any algorithm based on R-trees.

Recently, numerous variations of skyline queries have been studied as well. Examples include, to name but a few, (i) *subspace* skyline computation [25,34,39]; (ii) *reverse* skyline query [12,15,28]; (iii) *metric* skyline retrieval [9,13]; (iv) *continuous* skyline query [19,23]; (v) *distributed* skyline retrieval [8,18,40]; (vi) *uncertain* skyline query [33,47]; and (vii) skyline computation on *data streams* [29,35,38] and *incomplete data* [16], respectively.

However, the skyline operator may return too many skyline objects to offer any meaningful insights. To address this, several efforts on the skyline size control have also been made, by *relaxing the dominance relationship* or *integrating user-specific preference*. Specifically, Koltun and Papadimitriou [20] introduce *approximately dominating representatives*, a refinement of the skyline query that remedies the output volume problem at a small (and controlled) loss of accuracy. Zhang et al. [46] aim to find *strong skyline objects* in high dimensional spaces, which is the union of the skyline objects in all δ-*subspace* (of a space) that contains the

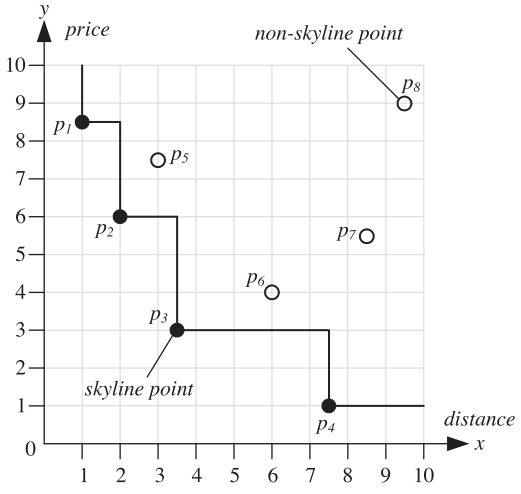

Fig. 1. Example of dataset and skyline.