Efficient Processing of Skyline Queries with Partially-Ordered Domains

Chee-Yong Chan
National University of Singapore
chancy@comp.nus.edu.sg

Pin-Kwang Eng
National University of Singapore
engpk@comp.nus.edu.sg

Kian-Lee Tan
National University of Singapore
tankl@comp.nus.edu.sg

1. Introduction

Many decision support applications are characterized by several features: (1) the query is typically based on multiple criteria; (2) there is no single optimal answer (or answer set); (3) because of (2), users typically look for *satisfying* answers; (4) for the same query, different users, dictated by their personal preferences, may find different answers meeting their needs. As such, it is important for the DBMS to present *all interesting* answers that may fulfill a user's need. In this paper, we focus on the set of interesting answers called the *skyline*. Given a set of points, the skyline comprises the points that are not *dominated* by other points [1]. A point dominates another point if it is *as good or better in all dimensions and better in at least one dimension*. As an example, a tourist looking for *budget* hotels that are *close* to the cities may issue the following SQL queries [1]: `Select * From hotels Skyline of Price Min, Distance Min`, where `Min` indicates that the price and the distance should be minimized. Clearly, if hotel h1 dominates hotel h2 (i.e., h1 is cheaper and nearer to the city than hotel h2), then h2 can be pruned away.

While much work has been done to develop efficient schemes to evaluate skyline queries, these deal exclusively with totally-ordered attribute domains [1, 4, 2, 3]. Partially-ordered attribute domains which include interval data (e.g., temporal data), categorical data (e.g., type/class hierarchies), and set-valued domains, have not been considered. In our hotel example, a hotel may store a set of interesting places/amenities within its vicinity, and our tourist may prefer a hotel that contains a larger set of interesting places/amenities (e.g., gift shop, gymnasium, saloon, sauna, etc.).

For totally-ordered attribute domains, index-based algorithms like NN algorithm [2] and BBS algorithm [3] have been shown to be superior over the nested-loop approach. However, because of the lack of a total ordering for partially-ordered attribute domains, it is unclear if index-based schemes can still maintain their competitiveness given that their effectiveness to prune the search space are reduced.

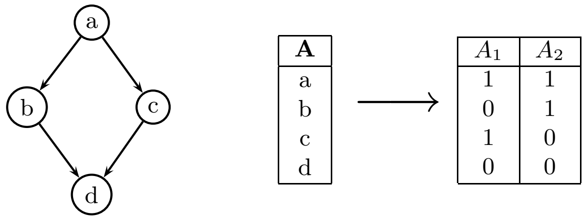
In this paper, we address the novel and important problem of evaluating skyline queries involving partially-ordered attribute domains. To the best of our knowledge, this issue has not been investigated by any of the previous related work.

2. Motivation

In this section, we consider the possible evaluation strategies and motivate our proposed algorithms for processing skyline queries with partially-ordered attribute domains. For convenience, we refer to such queries as *partially-ordered skyline queries* (or POS-queries) in contrast to the *totally-ordered skyline queries* (or TOS-queries) that involve only totally-ordered attribute domains.

The most direct method to process POS-queries is to apply the well-known block nested loop approach (BNL) [1], which is the simplest and most versatile approach that works for all types of attribute domains. However, the performance of BNL has been shown to be inferior to that of index-based approaches (such as NN algorithm [2] and BBS algorithm [3]) due to the pruning effectiveness of index-based methods. Another limitation of BNL is that it is a "blocking" algorithm and lacks progressiveness (i.e., answers can only returned after all skylines are computed).

Another strategy to evaluate POS-queries is to try to leverage the effectiveness of previous index-based approaches for TOS-queries by first transforming the partially-ordered attribute domains into totally-ordered domains such that the partial ordering of the original domains are "preserved" in the transformed domains. The most obvious transformation technique is to map each partially-ordered attribute domain into a set of boolean attribute domains as illustrated by the simple example in Fig. 1, where the attribute A (with partially-ordered do-



A
a
b
c
d

$\longrightarrow$

A_1	A_2
1	1
0	1
1	0
0	0

Figure 1. Example of domain transformation

main values $\{a, b, c, d\}$) is mapped into two boolean attributes A_1 and A_2. In this way, the collection of transformed attributes is now amenable to be indexed using one of the efficient techniques proposed for TOS-queries (e.g., [2, 3]). This transformation is particularly convenient for set-valued attribute domains. However, this approach suffers from the well-known "dimensionality curse" problem when the size of the partially-ordered attribute domain is large, which will be transformed to a large number of boolean-valued attributes. Thus, the simple boolean mapping is not suitable for index-based methods.

3. Our Approach

To both enable the use of efficient index-based techniques (that are designed for totally-ordered attributes) as well as avoid the "dimensionality curse" problem with using simple domain transformations, the approach that we propose is a "middle-ground" solution that is based on using an approximate, space-efficient domain transformation. Our approach is based on using an approximate *interval* representation (in the form of a pair of integer attributes) for each partially-ordered attribute. This strategy, which increases the dimensionality by one for each partially-ordered attribute, provides a reasonable and practical approximate domain mapping that is amenable to efficient indexing. Thus, the skyline answers can be computed by organizing the transformed attributes using an existing indexing method. Note that as the skyline computation is performed on the transformed space, false positives may arise and these have to be pruned away when answering skyline queries.

Based on the above framework, we propose three evaluation algorithms. BBS$^+$ is a straightforward adaptation of BBS. Because of false positives, BBS$^+$ is no longer *progressive*, i.e., it needs to find all skyline points before answers can be returned. The second scheme, SDC (Stratification by Dominance Classification) exploits the properties of domain mappings to avoid unnecessary dominance checkings. In particular, it organizes the data into two *strata* at runtime - points that are definitely in the skyline (stratum 1) and those that may be false positives (stratum 2) . As such, it can return answers in the former category as soon as they are produced. In the third scheme, SDC$^+$, the data is partitioned into two or more strata offline so that points at stratum i cannot dominate points at stratum $i - 1$. In this way, skyline points obtained from stratum $i - 1$ can be returned before points in stratum i are examined, thereby further improving on the progressiveness of the skyline computation.

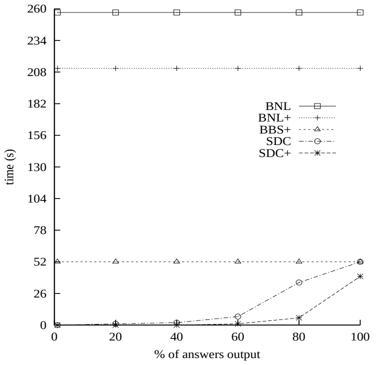


Figure 2. Performance Comparison

Fig. 2 compares the performance of our proposed algorithms (BBS$^+$, SDC, and SDC$^+$) against the block nested-loop algorithms (BNL and BNL$^+$) for a 3-dimensional dataset (with one partially-ordered atttribte) consisting of 500K records. SDC$^+$ has the best performance in terms of both response time and progressiveness.

4. Conclusions

In this paper, we addressed the problem of evaluating skyline queries with partially-ordered domains. Our proposed algorithms, which are based on using approximate domain transformations, outperformed existing approaches by a wide margin.

References

[1] S. Börzsönyi, D. Kossmann, and K. Stocker. The skyline operator. In *ICDE'01*, pages 421–430, 2001.

[2] D. Kossmann, F. Ramsak, and S. Rost. Shooting stars in the sky: an online algorithm for skyline queries. In *VLDB'02*, 2002.

[3] D. Papadias, Y. Tao, G. Fu, and B. Seeger. An optimal and progressive algorithm for skyline queries. In *SIGMOD'03*, pages 467–478, 2003.

[4] K. L. Tan, P. K. Eng, and B. C. Ooi. Efficient progressive skyline computation. In *VLDB'01*, pages 301–310, 2001.