An I/O Optimal and Scalable Skyline Query Algorithm

Yunjun Gao, Gencai Chen, Ling Chen, and Chun Chen

College of Computer Science, Zhejiang University, Hangzhou, 310027, P.R. China
{gaoyj, chengc, lingchen, chenc}@cs.zju.edu.cn

Abstract. Given a set of $d$-dimensional points, skyline query returns the points that are not dominated by any other point on all dimensions. Currently, BBS (branch-and-bound skyline) is the most efficient skyline processing method over static data in a centralized setting. Although BBS has some desirable features (e.g., I/O optimal and flexibility), it requires large main-memory consumption. In this paper, we present an improved skyline computation algorithm based on best-first nearest neighbor search, called IBBS, which captures the optimal I/O and less memory space (i.e., IBBS visits and stores only those entries that contribute to the final skyline). Its core enables several effective pruning strategies to discard non-qualifying entries. Extensive experimental evaluations show that IBBS outperforms BBS in both scalability and efficiency for most cases, especially in low dimensions.

1 Introduction

Skyline query is one of important operations for several applications involving multi-criteria decision making, and has received considerable attention in the database community. Given a set of $d$-dimensional points $P = \{p_1, p_2, \ldots, p_n\}$, the operator returns a set of points $p_i$, which is not dominated by any other point $p_j$ in $P$ on all dimensions, forming the skyline of $P$. A point dominates another one if it is as good or better in all dimensions and better in at least one dimension [19]. Consider, for instance, a common example in the literature, “choosing a set of hotels that is closer to the beach and cheaper than any other hotel in distance and price attributes respectively from the database system at your travel agents’ [3]”. Figure 1(a) illustrates this case in 2-dimensional space, where each point corresponds to a hotel record. The room price of a hotel is represented as the $x$-axis, and the $y$-axis specifies its distance to the beach. Clearly, the most interesting hotels are the ones $\{a, g, i, n\}$, called skyline, for which there is no any other hotel in $\{a, b, \ldots, m, n\}$ that is better on both dimensions. For simplicity, in this paper, we use the min condition on all dimensions to compute the skyline, even though the proposed algorithm can be easily applied to different conditions (e.g., max metric). Using the min condition, a point $p$ is said to dominate another one $q$ if (i) $p$ is not larger than $q$ in all dimensions, and (ii) $p$ is strictly smaller than $q$ in at least one dimension. This implies that $p$ is preferable to $q$ for the users in real life. Continuing the running example,
hotel a dominates hotels b, d, and e because the former is nearer to the beach and cheaper than the latter.

Skyline query processing has been extensively studied, and a large number of algorithms have been also proposed [1, 3, 5, 7, 9, 12, 13, 15, 16, 19, 21]. These methods can be mainly divided into two categories. Specifically, (i) non-index-structure-based schemes, which do not assume any index structure on the underlying dataset, but compute the skyline through scanning the entire dataset at least once, resulting in expensive CPU overhead; (ii) index-structure-based solutions, which significantly reduce CPU and I/O costs by performing the skyline retrieval on an appropriate index structure (e.g., R*-tree [2]). We concentrate on the second category in this paper. In addition, the skyline computation problem is also closely related to several other well-known problems that have been extensively investigated in the literature, such as convex hull [4, 17], top-k queries [6, 8, 11, 14], and nearest neighbor search [10, 18].

Currently, BBS (branch-and-bound skyline), presented by Papadias et al. in [15], is the most efficient skyline query algorithm over static datasets in a centralized setting. It employs a best-first based nearest neighbor search paradigm on dataset indexed by R*-tree. BBS minimizes the I/O overhead, and the considerable experiments of [15] show that it outperforms previous algorithms in terms of CPU and I/O costs for all problem instances. Although BBS has some desirable advantages, it yet needs large memory space. As reported in [15], the heap size of BBS is larger than the to-do list size of NN [12] in 2-dimensional space. In fact, we can greatly reduce space consumption used by the heap and speed up the execution of the algorithm via several dominance checking based pruning heuristics (discussed detailedly in Section 3 of this paper) for filtering all the non-qualifying entries that may not contain (become) any skyline point. As known, the less the memory space requires, the more scalable the algorithm is. Thus, in this paper, we present an improved skyline query algorithm, called IBBS, which, like BBS, is depended on best-first nearest neighbor search on R*-tree, whereas (unlike BBS) it enables several effective pruning strategies to discard unnecessary entries. IBBS incorporates the virtues of BBS (e.g., I/O optimal, low CPU cost, etc.), while gaining less main-memory consumption (i.e., smaller heap size). Finally, extensive experiments with synthetic datasets confirm that IBBS outperforms BBS in both efficiency and scalability for most cases, especially in low dimensions.

The rest of the paper is organized as follows. Section 2 reviews existing algorithms for skyline queries, focusing more on BBS as it is more related to our