

On Skylining with Flexible Dominance Relation

Tian Xia Donghui Zhang
College of Computer and Information Science
Northeastern University
360 Huntington Avenue, Boston, MA 02115
{tianxia, donghui}@ccs.neu.edu

Yufei Tao
Dept. of Computer Science and Engineering
Chinese University of Hong Kong
New Territories, Hong Kong
taoyf@cse.cuhk.edu.hk

Abstract— Given a set of d dimensional objects, a skyline query finds the objects (“skyline”) that are not dominated by others. However, skylines do not always provide useful query results to users, and existing methods of various skyline queries have at least one of the following drawbacks: (1) the size of skyline objects can not be controlled, or can be only increased or only decreased but not both; (2) skyline objects do not have built-in ranks; (3) skylines do not reflect users’ weights (preferences) at different dimensions. In this paper, we propose a unified approach, the ε -skyline, to effectively solve all three drawbacks. We explore the properties of ε -skylines and propose several different algorithms to compute ε -skylines.

I. INTRODUCTION

The skyline query has recently attracted considerable attention due to its importance in many applications such as multi-criteria decision making [1], data mining [2], and user-preference queries [3]. Given a set of d dimensional objects T , an object t_1 is said to *dominate* another object t_2 if t_1 is better than or equal to t_2 in all dimensions, and is better than t_2 in at least one dimension. The *skyline* of T is the set of all objects in T that are *not* dominated by any other objects, and the *Skyline Query* is to return such skyline objects.

Skyline computation has been extensively studied (e.g. [1], [4], [5], [6]) in the past. However, skyline objects returned to users are not always satisfactory, due to three inherent drawbacks of the skyline definition. First, users have no way of knowing how large the returned skyline could be, and are unable to control the size of the skyline. Sometimes users get too few skyline objects to choose from, and sometimes they get an overwhelming number of skyline objects. Second, because of the nature of multi-criteria queries, skyline objects are incomparable to each other by definition, and do not have a semantic order. Here, we only consider built-in ranks among objects without relying on user-specified functions, since it is not always possible for users to provide such functions. Ranking the skyline objects is particularly important in high-dimensional spaces, where skyline sizes are usually very large and enumerating skyline objects with a meaningful order is desirable. Third, skylines are invariant to dimensional scaling. That is, shrinking or expanding each dimension will not change skylines. It is not unusual for users to want to scale some dimensions to show their preferences (e.g. place a weight of 30% on distance and 80% on hotel prices). Such user preferences cannot be reflected in conventional skylines, since users always get the same set of results with or without weights.

Motivated by these problems, in this paper we propose a unified approach, the ε -skyline, to overcome all three mentioned drawbacks of the conventional skylines. In particular, we define a flexible ε -dominance relation. An object t is said to ε -dominate t' if and only if t is smaller than or equal to t' plus ε in all dimensions, and is strictly smaller than t' in at least one dimension. The ε -dominance relation provides better semantics in relaxing the traditional skyline and an easy way to control skyline objects. In fact, the ε -skyline can *monotonically* vary from the empty set to the whole dataset, and

the conventional skyline is a special case of ε -skylines when $\varepsilon = 0$. Moreover, the ε -skyline can be altered by user weights. We also show that ε -skyline objects have a built-in order to support top- k skyline queries. As a summary, our key contributions are:

- 1) We propose a unified and semantically-strong solution, the ε -skyline with flexible dominance relationship, to overcome the shortcomings of the conventional skyline.
- 2) We theoretically explore the important properties of ε -skylines, and propose two carefully designed and efficient algorithms for the ε -skyline query.
- 3) Our experiments on both synthetic and real data verify the efficiency of our algorithms, and reveals interesting properties of ε -skylines.

The rest of the paper is organized as follows. Section II defines the ε -skyline, and discuss the properties. Section III presents the algorithms to the ε -skyline queries. Section IV review previous work on skyline computation and Section V concludes this paper.

II. THE ε -SKYLINE

Although there are several existing skyline variants addressing one or some of the skyline drawbacks we mentioned in Section I, they have some serious shortcomings. For example, the *thick* skylines [2] can be used only to *increase* the number of returned skyline objects, and the k -dominant skylines [7] is proposed only to *decrease* the number of returned skyline objects. [8] and [9] focused only on *ordering* skyline objects. Thus, a user wishing to control skyline sizes in *both directions* (i.e. increase and decrease) has to use two completely different systems.

We propose a comprehensive and unified approach, the ε -Skyline, to overcome all the drawbacks of the conventional skylines. Without loss of generality, in the rest of this paper we assume each dimension of the data space is normalized to $[0, 1]$. Formally, the d dimensional ε -skyline is defined as follows.

Definition 2.1: (ε -Skyline) Given a set of d -dimensional objects T with weights $W = \{w_i \mid i \in [1, d], 0 < w_i \leq 1\}$, and a constant $\varepsilon \in [-1, 1]$, for any two objects $t_1 = \{t_1[1], \dots, t_1[d]\}$ and $t_2 = \{t_2[1], \dots, t_2[d]\}$, if $\forall i \in [1, d], t_1[i] \cdot w_i \leq t_2[i] \cdot w_i + \varepsilon$, and $\exists j \in [1, d], t_1[j] < t_2[j]$, then t_1 is said to ε -dominate t_2 , denoted as $t_1 \prec_\varepsilon t_2$. The ε -skyline of the dataset is the set of all objects in T that are *not* ε -dominated by any other objects.

The ε -skyline can be smaller or larger than the conventional skyline, when ε is positive or negative, respectively. Conventional skylines are special cases of ε -skylines when $\varepsilon = 0$. Given a 2-dimensional dataset in Fig. 1, the conventional skyline consists of t_1 , t_2 and t_3 . Fig. 2 shows two examples of ε -skylines. Suppose $w_1 = w_2 = 1$. When $\varepsilon = 0.1$, the ε -skyline contains only t_2 and t_3 . The conventional skyline object t_1 is eliminated, since t_2 now ε -dominates t_1 . Formally, we have $t_2[X] \cdot 1 = 0.4 \leq 0.45 = t_1[X] \cdot 1 + 0.1$, $t_2[Y] \cdot 1 = 0.4 \leq 0.9 = t_1[Y] \cdot 1 + 0.1$, and $t_2[Y] \cdot 1 = 0.4 < 0.8 = t_1[Y] \cdot 1$. As shown in Fig. 2(a), the ε -dominant region of t_2 when $\varepsilon > 0$ can be visually drawn as if t_2 is moved on both the X and Y axis by ε to t'_2 . On the other

hand, when $\varepsilon = -0.1$, the non-skyline object t_4 is now included in the ε -skyline, since t_2 no longer ε -dominates t_4 . Formally, we have $t_2[X] \cdot 1 = 0.4 > 0.35 = t_4[X] \cdot 1 - 0.1$. The shaded region in Fig. 2(b) shows the ε -skyline region, which includes all four objects.

	t_1	t_2	t_3	t_4
X	0.35	0.4	0.7	0.45
Y	0.8	0.4	0.1	0.7

Fig. 1. Example of a 2-dimensional dataset.

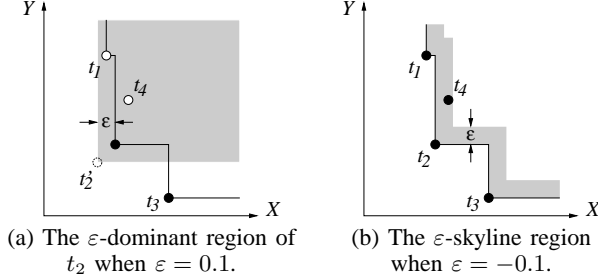


Fig. 2. Illustrations of ε -skylines.

The ε -skyline naturally incorporates user’s weights, and the set of ε -skyline objects may change in accordance with different weights on each dimension, reflecting users’s preferences. In the conventional skyline definition, any (positive) weight can be removed from both sides of comparison equations without affecting the correctness. However, in ε -skylines, weights cannot be omitted and may influence ε -skyline results.

There are several important properties of ε -skylines. While ε -skylines still hold *irreflexivity* (Property 2.2), the (strict) transitivity and (strict) asymmetry are broken. The irreflexivity property means one object does not ε -dominate itself, due to the “strictly better” condition in the definition. In fact, when $\varepsilon \leq 0$ (which includes conventional skylines), transitivity and asymmetry are still preserved. However, when $\varepsilon > 0$, both properties may be broken. For example, consider the dataset in Fig. 1. If $\varepsilon = 0.3$, t_2 and t_3 are mutually ε -dominated by each other, and the asymmetry does not hold. Nevertheless, the ε -dominance relation holds *loose transitivity* (Property 2.3) and *loose asymmetry* (Property 2.4).

Property 2.2: (Irreflexivity) $\forall t, t \not\prec_\varepsilon t$.

Property 2.3: (Loose transitivity) $\forall t_1, t_2, t_3, (t_1 \prec_\varepsilon t_2 \wedge t_2 \prec_\varepsilon t_3) \vee (t_1 \prec_\varepsilon t_2 \wedge t_2 \prec_\varepsilon t_3) \Rightarrow t_1 \prec_\varepsilon t_3$.

Property 2.4: (Loose asymmetry) $\forall t_1, t_2, t_1 \prec_\varepsilon t_2 \Rightarrow t_2 \not\prec_\varepsilon t_1$, and $t_1 \prec_\varepsilon t_2 \Rightarrow t_2 \not\prec_\varepsilon t_1$

Next, Theorem 2.5 shows three special cases of the ε -skyline: it can be an empty set, the conventional skyline or the whole dataset.

Theorem 2.5: Given a set of d -dimensional objects T with weights W , let S_c be the conventional skyline of T . If $|S_c| > 1$, then $S_{\varepsilon=-1} = T$, $S_{\varepsilon=0} = S_c$, and $S_{\varepsilon=1} = \emptyset$. If $|S_c| = 1$, then $S_{\varepsilon=-1} = T$, $S_{\varepsilon \geq 0} = S_c$.

Furthermore, Theorem 2.6 shows that the set of ε -skyline objects vary *monotonically* from empty set to the whole dataset with the decrease of ε . With an appropriate ε , the ε -skyline with desired size can be returned to users.

Theorem 2.6: (Monotone) Given a set of d -dimensional objects T with weights W , two constants $\varepsilon_1, \varepsilon_2 \in [-1, 1]$, let S_{ε_1} be the ε_1 -skyline of the dataset, and S_{ε_2} be the ε_2 -skyline. If $\varepsilon_1 > \varepsilon_2$, $S_{\varepsilon_1} \subseteq S_{\varepsilon_2}$.

III. COMPUTING ε -SKYLINES

In this section, we discuss how to compute ε -skylines. Since the ε -skyline may not have strict transitivity and asymmetry, the existing skyline algorithm are ineligible to solve ε -skyline queries. In

addition, when designing the algorithms, we also keep in mind that our algorithms are not in the same category, so that they cover a wide range of scenarios in real applications. In Section III-A, we proposed a simple yet effective algorithm, ε -SFS, which extends the counterpart in conventional skylines. In Section III-B, we propose an Index-based Filter-Refinement (IFR) algorithm, which utilizes existing spatial indices (e.g. R-trees) and the filter-refinement framework.

A. The ε -SFS Algorithm

Due to the lack of strict transitivity and asymmetry in ε -skylines, directly applying the conventional skyline algorithm SFS will incur two problems when $\varepsilon > 0$. First, if an object t is ε -dominated by some object, t cannot be immediately discarded, since it may be used to prune latter objects. Second, if an object t is not ε -dominated by any existing skyline objects, t cannot be immediately reported as a skyline object. Because it is possible that t may be ε -dominated by a latter object.

Our ε -SFS overcomes these two problems, and is still able to produce ε -skyline objects progressively. We first (pre-)sorts all objects $t \in T$ in ascending order of the monotone function $f(t) = \sum_{1 \leq i \leq d} t[i]/d$ - the average of (weighted) coordinate values. By Definition 2.1, we have Lemma 3.1.

Lemma 3.1: Given a set of d dimensional objects T with weights W , $\forall t_1, t_2$, if $t_1 \prec_\varepsilon t_2$, then $f(t_1) < f(t_2) + \varepsilon$.

The implication of Lemma 3.1 is twofold. First, when $\varepsilon > 0$, a candidate ε -skyline object t_c can be reported only if the next object to be examined has $f(t) \geq f(t_c) + \varepsilon$. This is because t and its successors (with even larger $f(\cdot)$) cannot ε -dominate t_c . Second, recall that SFS needs to compare a new object with all filter objects to make sure it is in the skyline. In ε -SFS, however, this is not necessary. When ε is negative, by Lemma 3.1, one object does not need to be compared to the filter objects with $f(\cdot) \geq f(t)$, as they cannot ε -dominate t .

B. The Index-based Filter-Refinement Solution

In this section, we present a novel approach based on a spatial index. Taking advantage of the existing data-partitioning spatial index built on the dataset, we may achieve more pruning power than the previous solution. Our IFR algorithm adopts the commonly-used filter-refinement framework, and uses R-tree for its popularity.

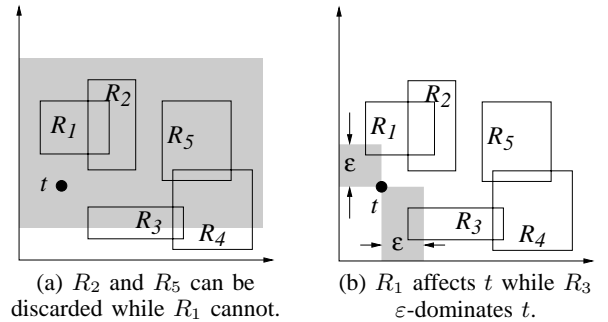


Fig. 3. Illustration of different cases among MBRs and a candidate.

One challenge in ε -skylines ($\varepsilon > 0$) is that relationships among objects and MBRs are much more complex. Consider the example in Fig. 3 (a). Each rectangles is the minimum bounding rectangle (MBR) of all objects inside it. The shaded region is the ε -dominant region of t . Although MBRs R_1, R_2 and R_5 are all ε -dominated by t , they should be treated differently. Since the lower left corner of R_2 and R_5 are strictly dominated by t , all objects in the two MBRs are also strictly dominated by t . Based on the loose transitivity (Property 2.3), R_2 and R_5 can be safely pruned. However, R_1 may not be pruned as it may contain some object ε -dominating t . Fig. 3(b) visually illustrates how R_1 affects t . The shaded region represents the *affecting region* of t . It can be proved that any object in the affecting

region will ε -dominate t . Since R_1 intersects with the affecting region of t , R_1 may contain some object that falls into the overlapping region and ε -dominates t .

Furthermore, we identify another important case in Fig. 3(b). Without expanding R_3 , we guarantee that R_3 must contain at least one object ε -dominating t . This is because R_3 has an entire edge (or a $d - 1$ hyperplane for arbitrary d) contained in the affecting region of t . For any minimum bounding rectangle, there must exist at least one object on each $d - 1$ hyperplane of an MBR, which is contained in the affecting region. Thus t can be eliminated without expanding any of its nearby MBRs.

Straightforwardly expanding every MBR that affects a candidate is not efficient. For example, in Fig. 3, suppose R_1 is visited before R_3 . Although R_1 affects t , it should be not expanded as t is later filtered as a false positive by R_3 . Therefore, in our IFR, we maintain an extra set *ToExpand* for MBRs like R_1 and use a *recursive refinement function* to efficiently eliminate false positives.

Our algorithm follows the filter-refinement paradigm. A set *ToExpand* of entries is kept to be expanded in the refinement step. The filter step integrates the above considerations into the branch-and-bound routine. For example, when an entry e is ε -dominated by some existing object, e is put into *ToExpand* set. Since e cannot contain any candidate, e is not expanded in the filter step, while e may be used to remove false positives in the refinement step. The refinement step recursively removes false positives using *ToExpand*. We associate each entry in *ToExpand* with all affected candidates. Then we recursively expand each entry affecting at least one candidate. There is a heuristic of which entry should be expanded first. We choose to expand the entry that affects the most candidates. The rationale is candidates may be eliminated in previous expansions, and latter entries with fewer associations may not need to be expanded at all.

C. Experimental Comparison

We compare our two algorithms using the NBA data, which are NBA player statistics from 1946 to 2004 (<http://databaseBasketball.com>), with 16,377 records. We choose 7 statistics as attributes that are not blank in all these years (e.g. points, assists, and etc). Fig. 4 shows the effects of ε on ε -skyline queries. In general, when $\varepsilon (> 0)$ increases (Fig. 4a), more objects are ε -dominated. Thus the running time of both algorithms decreases. When $\varepsilon (< 0)$ decreases (Fig. 4b), we will retrieve more objects as ε -skylines, and both algorithms' costs increase too. It can be seen that IFR is more sensitive to the ε than ε -SFS.

When $\varepsilon (< 0)$ decreases, one interesting case in Fig. 4(b) is that ε -SFS actually runs faster despite the fact that the number of ε -skyline object increases dramatically. This is due to one of our feature in ε -SFS. If the differences are small, with large ε most objects are directly reported as ε -skyline objects without comparison, as the existing objects cannot ε -dominate them by Lemma 3.1.

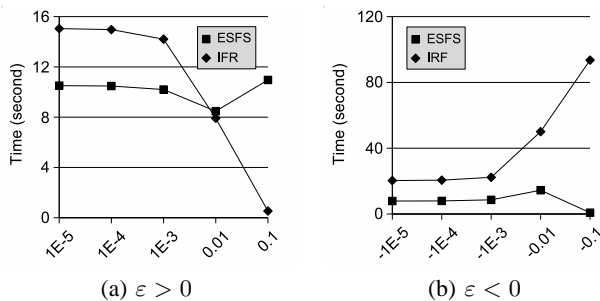


Fig. 4. Effects of ε on ε -skyline queries.

IV. RELATED WORK

Börzsönyi *et al.* [1] introduced the skyline operator in the database context, and proposed the *Block Nested Loop (BNL)* algorithm for

skyline queries. Chomicki *et al.* [4] proposed SFS (sort filter skyline), which extends BNL by pre-sorting data on a monotone function. The advantages of SFS over BNL are that SFS progressively outputs skyline objects and reduces the number of comparisons between objects. Recently, Godfrey *et al.* [5] proposed LESS (linear elimination sort for skyline) which eliminates non-skyline records during the pass zero of the external sort routine, and combines the last pass of the external sort with the first skyline filter pass.

The state-of-the-art index-based algorithm BBS (branch and bound skyline) performs a single traverse of the R-tree and visits only once the tree nodes which may contain skyline objects. Any object added into the intermediate skyline during the execution of BBS is guaranteed to be in the final skyline.

Jin *et al.* [2] proposed the thick skyline to increase the number of final results returned to users. The thick skyline also includes non-skyline objects within ε distance of each skyline object. These non-skyline objects could provide users with more choices when the skyline objects are limited. Later, Chan *et al.* [7] proposed k -dominant skylines to decrease the number of skyline objects in high dimensional spaces. A d -dimensional object t is said to k -dominate ($k \leq d$) t' if t dominates t' in an arbitrary k dimensional subspace. With the decrease of k , the size of the k -dominant skyline becomes smaller. Both variants of the conventional skyline definition can only control the skyline size in one direction.

As to ranking skyline objects, there are several new concepts that have been proposed. Chan *et al.* [8] proposed the *skyline frequency* to rank the skylines, so that the skyline objects can be output in top- K fashion. The skyline frequency of object t is defined as the number of subspaces in which t is a skyline object. Lin *et al.* [9] investigated the problem of finding k representative skyline objects (*top-k RSP*) such that the number of objects being dominated by one of the k skyline objects is maximized.

Acknowledgement: Donghui Zhang's research has been partially supported by NSF CAREER Award IIS-0347600.

V. CONCLUSIONS

In this paper, we present a flexible and semantically-strong skyline concept, the ε -skyline. Our ε -skyline overcomes the drawbacks of conventional skylines and previous skyline variants, and achieves the following goals: (1) users are able to control the number of output skyline objects in both directions - increase and decrease; (2) the ε -skyline provides a built-in rank for all objects, to support top- k ε -skyline queries; and (3) the ε -skyline naturally integrates the weights on the dimensions, and is responsive to different weights. We also show that ε -skyline vary monotonically from empty set to the whole data set when ε decreases.

REFERENCES

- [1] S. Börzsönyi, D. Kossmann, and K. Stocker, "The Skyline Operator," in *ICDE*, 2001.
- [2] W. Jin, J. Han, and M. Ester, "Mining Thick Skylines over Large Databases," in *PKDD*, 2004.
- [3] V. Hristidis, N. Koudas, and Y. Papakonstantinou, "PREFER: A System for the Efficient Execution of Multi-parametric Ranked Queries," in *SIGMOD*, 2001.
- [4] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang, "Skyline with Presorting," in *ICDE*, 2003.
- [5] P. Godfrey, R. Shipley, and J. Gryz, "Maximal Vector Computation in Large Data Sets," in *VLDB*, 2005.
- [6] D. Papadias, Y. Tao, G. Fu, and B. Seeger, "An Optimal and Progressive Algorithm for Skyline Queries," in *SIGMOD*, 2003.
- [7] C. Y. Chan, H. V. Jagadish, K.-L. Tan, A. K. H. Tung, and Z. Zhang, "Finding k -dominant skylines in high dimensional space," in *SIGMOD*, 2006.
- [8] —, "On High Dimensional Skylines," in *EDBT*, 2006.
- [9] X. Lin, Y. Yuan, Q. Zhang, and Y. Zhang, "Selecting Stars: the k Most Representative Skyline Operator," in *ICDE*, 2007.