On the Suitability of Skyline Queries for Data Exploration

Sean Chester, Michael L. Mortensen, and Ira Assent
Data-Intensive Systems, Aarhus Universitet
Åbogade 34 8200-Århus N, Denmark
{schester, illio, ira}@cs.au.dk

ABSTRACT

The skyline operator has been studied in database research for multi-criteria decision making. Until now the focus has been on the efficiency or accuracy of single queries. In practice, however, users are increasingly confronted with unknown data collections, where precise query formulation proves difficult. Instead, users explore the data in a sequence of incrementally changing queries to the data to match their understanding of the data and task. In this work, we study the skyline operator as a tool in such exploratory querying both analytically and empirically. We show how its results evolve as users modify their queries, and suggest using our findings to guide users in formulating reasonable queries.

1. INTRODUCTION

Say you have never been to America and you find yourself in Manhattan searching for a restaurant. Where do you even begin? Probably, you want something close, but quite what is "close" may not be clear. If you might go to a show later, several locations can be equally valid reference points for "close." Perhaps you prefer something inexpensive, but having never been to Manhattan, what really is "expensive"? Search sites can help, but only if you know for what to look.

The *skyline* operator is said to be useful in this context, because it identifies the data points (restaurants) that express the best trade-offs between the dimensions of interest (proximity, rating, and price). But what if the user wants to *explore* the data, and may evolve new preferences throughout the process? He/she may decide that price, afterall, is no concern, or not to look at any more pizzerias. For the skyline to be useful in this interactive process, it is crucial that one can continually add constraints and change dimensions of interest without completely changing the results that he/she sees. If the skyline filters too many points that it did not filter before, the user will likely be as mystified as the users in the skyline user study of Magnani et al. [8].

An interactive skyline has been assumed in several contexts (e.g., skycube computation [5], dynamic skylines [4], visualization [8], anytime computation [9], and preference elicitation [1, 7]), but how the interaction affects the skyline is not well understood. If a small query changes produce radically different skylines, which is theoretically possible (Section 3), then an interactive skyline would not make sense.

Nevertheless, if the skyline accomodates making incremental changes to a query formulation, it has potential to help an exploratory user. So, in this paper, we take a first look at how suitable the skyline is when repeatedly executed on slightly different views of the data. In particular, we ask:

1. What can theoretically happen to the results of a skyline when a query is incrementally vs. arbitrarily reformulated (Section 3)?
2. Does being in one skyline make being in a skyline for similar queries more likely? Or is one query's result uninteresting once the user's preferences evolve?
3. How often are the theoretical effects in Section 3 empirically observed in real and synthetic data (Section 4)?

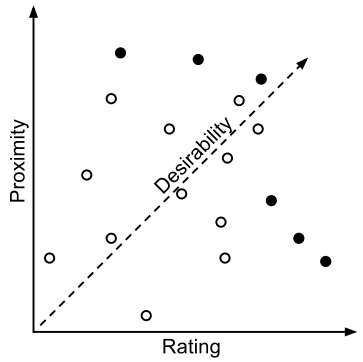


Figure 1: Example of a skyline. The black points are in the skyline because no other points have, relative to them, both a higher rating and proximity.

2. BACKGROUND

Figure 1 illustrates the skyline [2], a filtration tool. Given a set $\mathcal{D}$ of n points $p = (p_0, \ldots, p_{d-1})$ in d dimensions, the *skyline* consists of all points $p \in \mathcal{D}$ that are not dominated by any other points $q \in \mathcal{D}$. A point q dominates another point p if $\forall 0 \leq i < d, q_i \geq p_i$ *and* $\exists 0 \leq i < d, q_i > p_i$.[1] That is to say, for any pair of non-equal points, if one is in the skyline it must have a higher value than the other on

[1]The assumption of preferring larger values is WLOG: one can multiply any attribute by -1 in preprocessing.

some attribute. The skyline consists of all points that are not inferior to some other point.

However, the number of skyline points can be quite large [3] and a user may find only some dimensions and values to be interesting; so, the skyline operator should be combined with subspace projections (then called a *subspace skyline* [12]) and with range constraints (then called a *constrained skyline [10]*). Our general form of a skyline query is then:

SELECT <*subspace*>
FROM <*tables*>
WHERE <*range constraints*>
SKYLINE <*min/max specifications*>

A user specifies dimensions are of interest (the subspace), min and max values for those dimensions (range constraints), and whether he/she prefers smaller or larger values on each attribute (the specifications). To (logically) execute such a query, one first applies the constraints and projections, and then computes the skyline of the resultant, filtered dataset.

In an exploratory context, the skyline is not the terminus of the process. After observing the results, the user will reformulate the query to match evolved understanding *in an incremental manner*. That is to say, subsequent queries in an exploratory process are not disjoint, because if they were, that would imply that the user is completely dissatisfied with the results of the first search, since he/she deliberately excluded them from the second search. This is effectively restarting. We will focus on largely overlapping queries, which suggest some successful interactivity.

More precisely, we define an *incremental change* as additions *or* removals of subspace dimensions or an edit to one range constraint. Such incremental changes can produce query results similar to the ones before. Any larger changes can be decomposed into a sequence of incremental ones. However, while the points satisfying the constraints likely are similar after an *incremental change*, the extent to which the skyline changes is not well known.

Problem statement

Given a *baseline* query, consisting of a subspace projection and a set of range constraints, and an *incremental change* to that query, how many points do the skyline of the baseline and the skyline of the modified query have in common?

3. THEORETICAL EFFECTS

In this section, we look at what *can* happen to the results of a skyline query after an incremental change.

3.1 Effect of varying constraints

Consider a skyline query applied to some baseline constraints (Figure 2). The results are very specific to the constraints posed; for example, although points p and l would be part of the skyline if there were no constraints, neither match any user constraints. On the other hand, whereas g and h are not part of the unconstrained skyline, they become skyline points if l is eliminated by the constraints.

In fact, *every* point can be part of the skyline for *some* set of constraints and *no* point is guaranteed to always be a skyline point. Therefore, an arbitrary change in constraints can have unpredictable consequences to the skyline: possibly adding new skyline points, removing existing ones, or both.

What we show here, however, is that if a user makes only an *incremental change*, the behaviour is predictable. There are four *effects* that can be observed, two types of skyline point addition and two types of skyline point removal:

1. **Addition** (A): A point only satisfies the new constraints and so becomes a skyline point;
2. **Removal** (R): An existing skyline point only satisfies the old constraints and so is removed from the skyline;
3. **Promotion** (P): A point becomes a new skyline point because all the points that dominated it are removed by the new constraints;
4. **Demotion** (D): An existing skyline point is removed from the skyline because it becomes dominated by some point that only satisfies the new constraints, but not the old ones.

Although an arbitrary change to constraints can induce any or all effects, an incremental change cannot. On a given attribute, there can be both an upper (U) and a lower (L) constraint, either of which can be increased (I) or decreased (D). We analyze each of these four cases:

LD. In Figure 2(b), the lower constraint is decreased. This **adds** a to the skyline, which has a high y-value but had an x-value outside the constraints. An LD change to dimension D can add points to the skyline if they are in the subspace skyline on the remaining dimensions, but it can never **remove** or **promote** points. New points have lower D-values than, and so could never dominate, existing skyline points.

LI. In Figure 2(c), the lower constraint is increased. This **removes** d from the skyline: it no longer matches the constraints. LI changes to dimension D consider no new points (so cannot **add**) and remove those smallest on D. Their removal cannot result in a **promotion**, because points satisfying the new constraints must have a higher D-value so cannot have been previously dominated by the removed points.

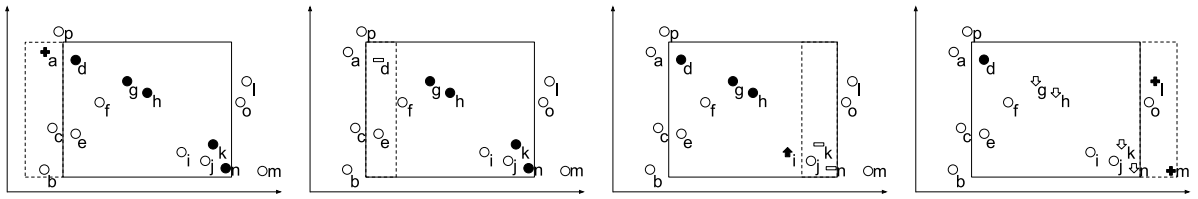
UD. In Figure 2(d), the upper constraint is decreased. The points n and k are **removed**, but point i, which was dominated by k, is **promoted**. The **add** effect cannot occur, because all points matching the new constraints matched the old constraints. Therefore, for a point q to become a new skyline point subject to the new constraints, the point that dominated q in the old constraints must be eliminated.

UI. In Figure 2(e), the upper constraint is increased. Consequently, points l and m are **added** and points $\{g, h, k, n\}$ are all **demoted** because they are dominated by l. Existing skyline points cannot be simply **removed**, because they necessarily match the new constraints: they can only be eliminated from the skyline by becoming dominated.

A user can control which of the effects could happen by making an incremental change (one constraint), e.g., LU to decrease the skyline size. Composing constraint modifications on different attributes is only predictable if the modifications are all of the same type. *How much* each of these effects is observed we evaluate empirically (Section 4).

3.2 Effect of varying subspace projections

Incremental changes to subspace projections can also create four effects. As with the constrained skylines, there are two types of skyline addition and two types of removal:



(a) Dec. lower const. (LD) (b) Inc. lower const. (LI) (c) Dec. upper const. (UD) (d) Inc. upper const. (UI)

Figure 2: The four incremental constraint changes. The solid rectangle shows the baseline constraints and the dotted lines indicate the modification to the constraints. Solid points are in the skyline and hollow points are not. A *plus* indicates an **addition**; a *minus*, a **removal**; an *upwards arrow*, a **promotion**; and a *downwards arrow*, a **demotion**.

1. **Addition** (A): A point dominated in the old subspace becomes *incomparable* to the points that dominated it, and thus becomes a skyline point in the new subspace;
2. **Removal** (R): A skyline point in the old subspace is now *dominated* in the new subspace, so no longer belongs to the skyline;
3. **Homogenization** (H): A point not in the skyline in the old subspace becomes a skyline point because it *is identical* to some skyline point in the new subspace;
4. **Differentiation** (D): A skyline point that *was identical* to another skyline point in the old subspace is dominated by that skyline point in the new subspace.

For an example, consider the dataset, $\mathcal{D}_{\text{ex}}$, below, and the incremental addition of subspace dimensions, from $\{x_0\}$ to $\{x_0, x_1\}$ to $\{x_0, x_1, x_2\}$ (i.e., just the first value of each point, then the first two, then all three).

$$\mathcal{D}_{\text{ex}} = \begin{cases} p = & (1,2,2) \\ q = & (1,1,2) \\ r = & (0,2,3) \end{cases}$$

To begin, the skyline in x_0 is $\{p, q\}$, since p and q both have higher values than r on x_0, but not than each other. By adding dimension x_1, the skyline becomes $\{p\}$, an instance of **differentiation**. Point q is removed from the skyline because it is no longer identical to point p, instead now dominated by it. Finally, adding the last dimension, the skyline becomes $\{p, r\}$, an instance of **addition**. Point r is added to the skyline because it has a higher value than the other skyline point, p, on x_2; so, they have become incomparable.

In the other direction, we observe the inverse effects, first the **removal** of r and then then **homogenization** of q.

So, whether one adds or removes a dimension, points can be both added or removed from the skyline. Therefore, theoretically at least, one cannot anticipate how the skyline might change going from one subspace to a neighbour without using sophisticated preprocessing techniques, such as those in [11]. So, we will determine the actual *frequency* of these effects empirically (Section 4).

A note about Distinct Value Condition

Effects H and D create unpredictability when changing subspaces. Thus the motivation for Distinct Value Condition [11], which ensures monotonicity. In particular, if no value appears twice in the dataset for the same attribute, then a point in the skyline for some subspace will always remain in the skyline after adding any number of other dimensions.

4. EMPIRICAL INVESTIGATION

In the previous section, we investigated what theoretically happens to the result of a skyline query if one makes incremental changes to the subspace or constraints. In this section, we investigate *how often* each of these effects empirically occur. Our strategy with these experiments is to execute an initial subspace or constrained query, modify the query formulation by a variable extent, and measure the occurrences for each effect defined in Section 3.

4.1 Setup

To observe incremental changes, we conduct one suite of experiments in which we adjust constraints (Section 4.2.1) and one in which we add dimensions to a subspace (Section 4.2.2). We briefly describe implementation details (Section 4.1.1), the datasets that we use (Section 4.1.2), and the methodology (Section 4.1.3).

4.1.1 Implementation Details

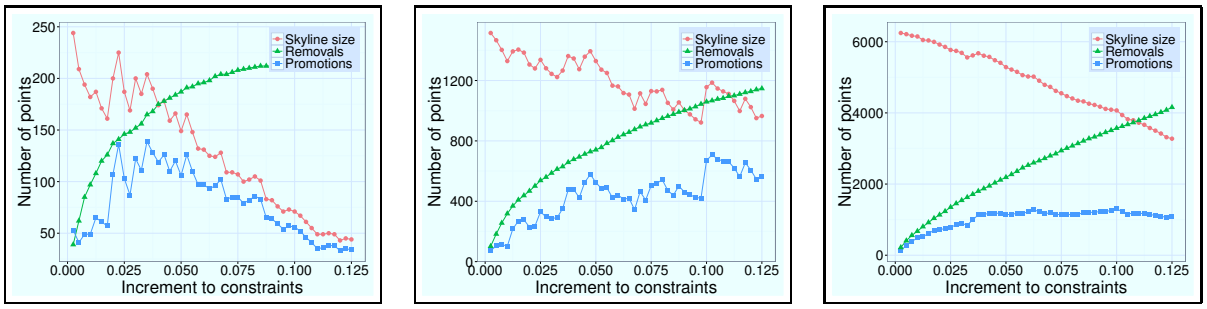
We first apply the contraints/subspace projections onto the data with a short *awk* program, and then apply a known skyline algorithm. When applying constraints, we use the state-of-the-art skyline algorithm, *BSkyTree* [6] (implemented by the original authors). As the *BSkyTree* algorithm does not handle duplicate points–which occur quite frequently in some subspaces–we use our own implementation of *BNL* [2] when applying subspace projections.

4.1.2 Datasets

For the experiments with constraints, we primarily use the standard skyline synthetic data generator [2],[2] with parameters we discuss in Section 4.1.3. The synthetic data permits drawing general conclusions with respect to the specific data distributions. To also observe behaviour on real data, we choose the *nba*[3] dataset, a standard benchmark for skyline research, consisting of statistics for 21961 basketball player-seasons. We use eight of the statistics, *gp*, *pts*, *asts*, *pf*, *fga*, *fgm*, *fta*, and *ftm*, because others contain frequent NULLs.

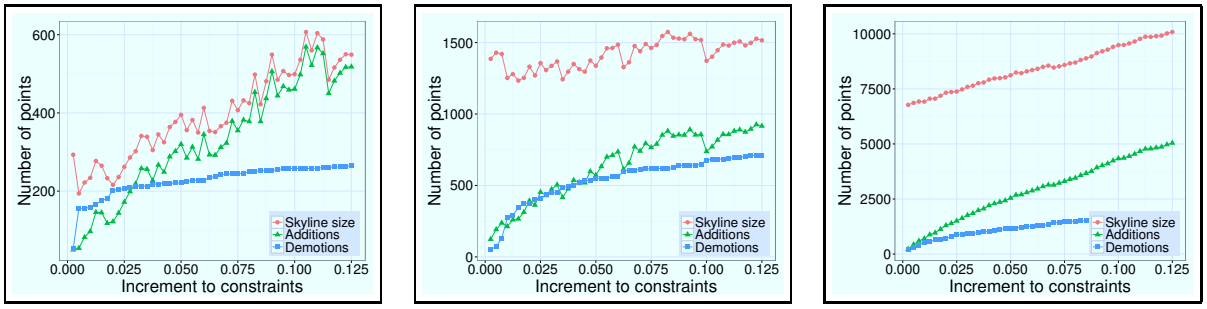
The behaviour between subspaces is only interesting without Distinct Value Condition (Section 3.2); so, we again use the *nba* dataset, which has duplicate values, but not the synthetic data, which does not. We add the *automobiles* dataset,[4] which has 406 points and 8 dimensions (although we only use the first seven, because *origin* is non-ordered).

[2] `http://http://pgfoundry.org/projects/randdataset`
[3] `http://www.databasebasketball.com`
[4] `http://stat-computing.org/dataexpo/1983.html`



(a) Correlated (b) Independent (c) Anti-correlated

Figure 3: Decreasing an upper constraint - *UD*. ($n = 100K$, $d = 6$)



(a) Correlated (b) Independent (c) Anti-correlated

Figure 4: Increasing an upper constraint - *UI*. ($n = 100K$, $d = 6$)

We choose this dataset to observe behaviour with many duplicate maximum values: 27% of the cars have the max. number (8) of cylinders, 7.5% of the cars have the max. year (1982), and 4.5% of the cars have the max. displacement (98.0). Non-maximal values are duplicated, too (e.g., 6 cylinders). All datasets are normalized to the range $[0, 1]$ to make the interpretation of our plots easier.

4.1.3 *Methodology*

There are many ways to reformulate a query, so many variables to consider empirically. We focus on those most natural for a user to tune and containing polar cases that demonstrate the range of skyline behaviour.

Studying constraints, we use synthetic data, introducing more variables but more generalizable findings. We hold the number of (and ergo density of) points constant at $100K$ and dimensionality at 6. As typical in skyline literature, we vary the distribution (correlated, indepedent, and anti-correlated), since the skyline size varies with correlation.

We pose a constant initial seed constraint that prunes 75% of the data (≥ 0.75 or ≤ 0.25). This is a reasonable first constraint, equivalent to asking only for restaurants with ratings of at least 4.0 (on a range of 1 to 5). We then compare the skyline result to that for constraints in 50 increments of 0.0025. We vary the direction of these changes (as per Section 3), both increasing and decreasing the constraint. We place the initial constraint in two different, extreme locations: one in the maximal direction (a lower constraint) and one in the minimal direction (an upper constraint). We do this for every dimension. In total, this produces 72 combinations on synthetic data, each for which we plot the 50 comparisons between the baseline and the modified constraint.

We do the same with the *nba* dataset to produce 48 combinations (not varying correlation, using all 8 attributes).

For the subspace investigation, there are fewer variables. As discussed above, we use only real datasets. We iterate each of the $(2^d - 2)$ proper, non-empty subspaces and, for each, compare the skyline result to that produced after adding 1 or 2 dimensions. We only *add* dimensions, because removal is symmetric. This produces 6050 and 1932 combinations for the *nba* and *automobiles* datasets, respectively.

4.2 Discussion

In this section, we describe the salient observations on skyline behaviour when adjusting constraints (Section 4.2.1) and adding dimensions to subspaces (Section 4.2.2).

4.2.1 *Effects of constraints*

We present here our findings on the effects of incremental constraint changes. We present, first, for upper constraint changes (*UD* and *UI*, in the terminology of Section 3.1), and, second, for lower constraint changes (*LD* and *LI*).

Upper constraints. Figures 3a-3c show the skyline size, along with the number of **removals** and **promotions** for a representative *UD* case on synthetic datasets. We do not show **additions** nor **demotions**, since, in agreement with Section 3.1, there are none.

In all distributions, we see that as the size of change increases, the **removals** increase steadily and the **promotions** vary throughout. This is especially apparent for correlated data, where about 80% of the original skyline is removed after a 0.065 decrease of the upper constraint and about 80 **promotions** occur after a slight change of 0.020.

This corresponds to a user, say, decreasing his/her budget for a restaurant, so the large and variable number of promotions implies he/she will see plenty new options when filtering out that which is most expensive. We also see the skyline size has a net decrease; so, as the user would expect, narrower ranges produce fewer results. Consequently, a *UD* change is appropriate for a user who wants to see *new* points without substantially changed the baseline constraints.

Figures 4a-4c show the skyline size, **additions** and **demotions** for the *UI* case, where a user is, say, increasing his/her budget. The overall trend is an increasing skyline size with only slight drops on account of **demotions**. Again, the correlated data shows large changes to the skyline even on very slight changes to constraints: minute changes induce over 100 **demotions**, creating an immediate local drop in skyline size. To an exploratory user most of the original results will seemingly be no longer valid.

Overall, on upper constraint modifications, we see dramatic changes for correlated data because the skyline will generally be located around the upper boundary of the data space for all of the correlated dimensions. Thus when one upper constraint is changed, it is likely to affect every skyline point and the change is more immediate than we see for other distributions. For all distributions, we see that small changes to constraints can yield significant changes to the skyline result. Assuming rational users want to predict the outcome of their input actions, a viable strategy for interactively using the skyline is to only make especially small changes to upper constraints. If changes are too large, the skyline may seem unpredictable and uncontrollable.

Lower constraints. We omit plots for the **additions** caused by an *LD* change and for the **removals** induced by an *LI* change for space. In agreement with Section 4.2.1, we only observe one type of effect for each of these changes and any **additions** or **removals** have a proportionate effect on the skyline size. The effect grows linearly with the size of the change in constraints. A user adjusting a lower constraint is probably trying to filter the results, because it is contrary to the direction of his/her preferences. The results confirm that this is a viable strategy: lowering (raising) the constraint increases (limits) the output.

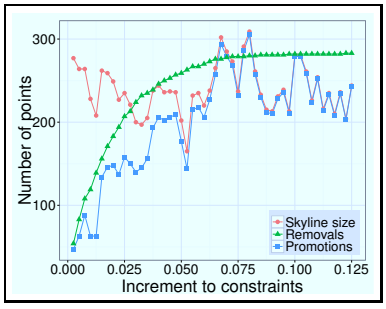


Figure 5: NBA - *UD*. ($n = 21961$, $d = 8$)

Real data. To investigate the effects from constraints on real data, we conducted the same experiments for the *nba* dataset. Figures 5 and 6 show the effects of decreasing the upper constraint on the *fgm* attribute and of increasing the upper constraint on the *fta* attribute, respectively. As was

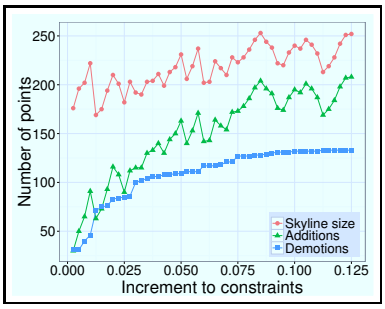


Figure 6: NBA - *UI*. ($n = 21961$, $d = 8$)

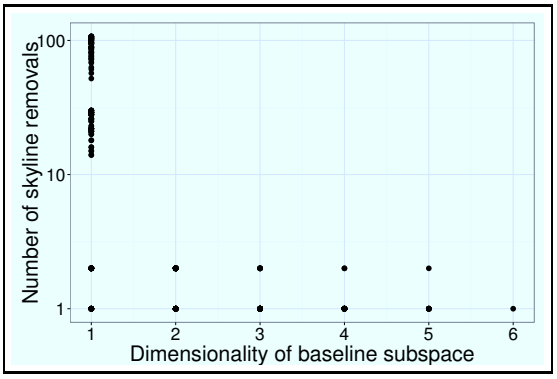


Figure 7: Scatterplot of skyline removals compared to dimensionality of the baseline subspace - *Automobiles*

the case with the synthetic data, we see a steady stream of **removals** and **demotions**. What is novel here is that while most of the skyline has been replaced after only a 0.04 decrease of the upper constraint in Figure 5 the skyline neither increases nor decreases heavily. The same trend is visible in Figure 6, where the skyline size has minimal variance.

This trend is interesting for an interactive analysis, since it shows a dataset like *nba* provides different skylines of comparable sizes, depending on the user's needs and expressed conveniently in the constraints. This further supports the strategy of using *UD* and *UI* cases to explore different skyline points in an incremental manner.

Results for the *LD* and *LI* cases are omitted due to space constraints, but confirm the trends shown in the synthetic experiments, supporting the strategy of using the *LD* and *LI* changes to regulate the skyline size.

4.2.2 Adding dimensions to subspace projections

In these experiments, we add dimensions to baseline subspace projections to observe how often the effects analyzed in Section 3.2 empirically occur. We only add dimensions, because the behaviour is symmetric (can be interpreted by reading the plots right-to-left) when removing dimensions. To a user, **Differentiations** (and **homogenizations**) are counter-intuitive, because they request more (less) data and then obtain fewer (more) results. So, we investigate under what circumstances this will impact the exploratory process.

On the *nba* dataset, we never observe **differentiations** for any configuration of experiment parameters. This is an interesting result because it shows that even in the presence of many duplicate values, one may not see any **differentiations** if those duplicates do not occur on the maximum

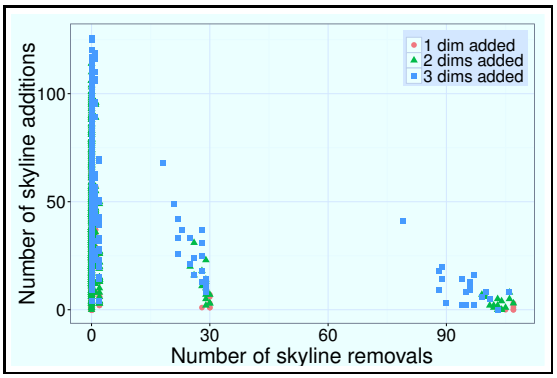


Figure 8: Scatterplot of skyline additions and removals caused by adding to baseline subspace - *Automobiles*

values. For the *nba* dataset, for example, there is seldom a tie for a record statistic such as *most points scored in a single NBA season*. So it is not surprising that *competitive* players do not have identical statistics in subspaces. Exploring the subspaces in this dataset will be quite intuitive.

On the *automobiles* dataset, we observe **diffentiations**. Figure 7 shows their frequency with respect to the the baseline subspace dimensionality when one dimension is added. Even with all dimensions, **differentiations** occur (points exist, albeit only 1). Beyond one dimension, all cases include the *cylinders* attribute, which has the most duplicated maximum values. On other attributes, there are no **differentiations** if the baseline has at least two dimensions. This illustrates that while **diffentiations** are rare when the baseline projection is on several dimensions, they do occur, and should be illustrated to the user so the results appear stable.

The second plot, Figure 8, shows how common are simultaneous **additions** and **differentiation**. The simultaneity is less desirable for exploring data, because it makes it harder to contrast subsequent queries. We see that if only one dimension is added (the triangular points), the effect can be predominantly **additions** or predominantly **differentiations** (on the axes), but not both (in the middle). This is because **differentiations** pre-suppose a high degree of homogeneity on maximal values, only one of which needs a high value on the new dimension in order to continue dominating all the points that are not in the baseline skyline. As the number of dimensions added goes up, there is a trend towards more mixed effects, because homogeneous points need to continue dominating non-baseline-skyline points over more new dimensions. By adding 3 dimensions, it is quite common to see roughly equal **additions** and **differentiations**; then, the result size has not changed (and thus is not easier to interpret), but the mixture of points has (which is counter-intuitive).

In summary, **differentiations** (**homogenizations**) are uncommon when adding (removing) one dimension, especially if starting with several dimensions. They are unlikely to occur at all if the duplicated values in the dataset are not on the maximal values. Nevertheless, they occur, even in high dimensional subspaces, and need to be illustrated to an exploratory user who would otherwise be confused.

5. CONCLUSION

In this work, we investigated how the skyline performs as a tool for exploratory data analysis. We analyzed how the skyline is affected by incremental changes with standard database operators (projection and selection) by defining the theoretical effects that one *can* observe, and measuring the frequency with which these effects *are* observed empirically.

A central motivation for the skyline is to disencumber the user from having to specify query parameters, and this research helps advance that objective. We envision that query recommendation can benefit from understanding the effects that incremental changes will have. If the goal is to produce new results, one can suggest some *UD/UI* changes or a dimension to remove. If the goal is to control the output size, *UD/UI* changes or additional dimensions can be automatically recommended. Future work can investigate strategies for producing these recommendations. Learning the consequences of manipulating query parameters is only a first step in exploring the expansive possibilities for how users and skyline-based systems can interact and in guiding exploration of new data in a principled manner.

6. ACKNOWLEDGEMENTS

This work has been supported in part by the Danish Council for Strategic Research, grant 10-092316.

7. REFERENCES

[1] W.-T. Balke, U. Güntzer, and C. Lofi. Eliciting matters – controlling skyline sizes by incremental integration of user preferences. In *DASFAA*, pages 551–562, 2007.

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

[3] C.-Y. Chan, H. V. Jagadish, K.-L. Tan, A. K. H. Tung, and Z. Zhang. On high dimensional skylines. In *EDBT*, pages 478–495, 2006.

[4] M. A. Cheema, X. Lin, W. Zhang, and Y. Zhang. A safe zone based approach for monitoring moving skyline queries. In *EDBT*, pages 275–286, 2013.

[5] J. Lee and S. Hwang. Qskycube: Efficient skycube computation using point-based space partitioning. *PVLDB*, 4(3):185–196, 2010.

[6] J. Lee and S. Hwang. Scalable skyline computation using a balanced pivot selection technique. *Information Systems*, 39:1–21, January 2014.

[7] J. Lee, G.-w. You, S. Hwang, J. Selke, and W.-T. Balke. Interactive skyline queries. *Information Sciences*, 211:18–35, 2012.

[8] M. Magnani, I. Assent, K. Hornbæk, M. R. Jakobsen, and K. F. Larsen. Skyview: a user evaluation of the skyline operator. In *CIKM*, pages 2249–2254, 2013.

[9] M. Magnani, I. Assent, and M. L. Mortensen. Anytime skyline query processing for interactive systems. In *DBRank*, 2012. No. 7.

[10] D. Papadias, Y. Tao, G. Fu, and B. Seeger. Progressive skyline computation in database systems. *TODS*, 30(1):41–82, 2005.

[11] J. Pei, Y. Yuan, X. Lin, W. Jin, M. Ester, Q. Liu, W. Wang, Y. Tao, J. X. Yu, and Q. Zhang. Towards multidimensional subspace skyline analysis. *TODS*, 31(4):1335–1381, 2006.

[12] Y. Tao, X. Xiao, and J. Pei. Subsky: Efficient computation of skylines in subspaces. In *ICDE*, page 65, 2006.