Using Multidimensional Skylines for Regret Minimization

Karim Alami and Sofian Maabout(✉)

Univ. Bordeaux, CNRS, LaBRI, UMR 5800, 33400 Talence, France
{karim.alami,sofian.maabout}@u-bordeaux.fr

Abstract. Skyline and Top-K operators are both multi-criteria preference queries. The advantage of one is a limitation of the other: Top-k requires a scoring function while skyline does not, and Top-k output size is exactly K objects while skyline's output can be the whole dataset. To cope with this state of affairs, regret minimization sets (RMS) whose output is bounded by K and where there is no need to provide a scoring function has been proposed in the literature. However, the computation of RMS on top of the whole dataset is time-consuming. Hence previous work proposed the Skyline set as a candidate set. While it guarantees the same output, it becomes of no benefit when it reaches the size of the whole dataset, e.g., with anticorrelated datasets and high dimensionality. In this paper we investigate the speedup provided by other skyline related candidate sets computed through the structure Negative SkyCube (NSC) such as Top k frequent skylines. We show that this query provides good candidate set for RMS algorithms. Moreover it can be used as an alternative to RMS algorithms as it provides interesting regret ratio.

1 Introduction

Skyline [3] and Top K [4] are two well known preference queries. The Skyline queries are based on the dominance relation. A tuple t is said to be dominated by a tuple t' iff (i) t' is *better* or equal on all dimensions and (ii) t' is strictly *better* on at least one dimension. The Skyline result is then the set of non dominated tuples. Top-K queries are based on scoring functions given users. Often scoring functions are linear, e.g., $f(t) = \sum_{i=1}^{d} w[i] * t[i]$ where w is called the weight vector. In a normalized setting, $0 \leq w[i] \leq 1 \; \forall i \in [1, d]$ and $\sum_{i=1}^{d} w[i] = 1$. The result of Top-K query by considering the scoring function f is K tuples with the best scores.

Example 1. Consider Table 1 that describes Hotels by their price and their distance from the beach. Suppose that cheaper and closer to the beach is better

The Skyline set with respect to this dataset is illustrated in Table 2. Only h_2 does not belong to the Skyline set because it is dominated by t_1. Indeed, t_1 is cheaper and closer to the beach

C. Attiogbé and S. Ben Yahia (Eds.): MEDI 2021, LNCS 12732, pp. 293–304, 2021.
https://doi.org/10.1007/978-3-030-78428-7_23

Table 1. Hotels.

Hotels	Price	Distance
h_1	200	120
h_2	390	140
h_3	465	20
h_4	395	90
h_5	100	300

Table 2. Skyline hotels.

Hotels	Price	Distance
h_1	200	120
h_3	465	20
h_4	395	90
h_5	100	300

Table 3. Top K Hotels.

Hotels – Weight vector	$(0.2, 0.8)$	$(0.5, 0.5)$	$(0.8, 0.2)$
h_1	136	$\underline{160}$	184
h_2	190	215	340
h_3	$\underline{109}$	242.5	376
h_4	151	242.5	334
h_5	260	200	$\underline{140}$

Table 3 represents the hotels' score wrt three linear scoring functions. Note that lower the value the better the hotel. Top-1 hotels score is underlined. h_1 is Top-1 wrt $(0.5, 0.5)$, h_3 is Top-1 wrt $(0.2, 0.8)$ and h_5 is Top-1 wrt $(0.8, 0.2)$

However Skyline queries and Top K queries have some limitations. On one hand, Skyline queries do not bound the results. Indeed the output may be the whole dataset, e.g., in presence of high dimensions and anti-correlated data. On the other hand, Top-K queries require the user to provide weight vector which is not an easy task. To solve these limitations, [8] presented the regret minimization queries. These queries bound the results and they do not require the user to provide the weight vector. Given a family of functions $\mathcal{F}$, they compute a subset $S \subset T$ that minimizes the maximum regret ratio. In a nutshell, the maximum regret ratio of a set S represents how far a user's best tuple in the whole dataset is from the best tuple in S. To simplify, consider the family of 3 functions $\mathcal{F} = \{f_{(0.2,0.8)}, f_{(0.5,0.5)}, f_{(0.8,0.2)}\}$ and consider a set $S = \{h_3, h_1\}$. The maximum regret ratio of S wrt $\mathcal{F}$, i.e., $mrr(S, \mathcal{F})$, is 31.4% which represents the ratio between the best score within T and the best score within S wrt the function

$f_{(0.8,0.2)}$. Concretely, this means that for a user whose scoring function is in $\mathcal{F}$, the best score he can get from S is at most 31.4% less than the best score he can get from T. Further details about the computation of the maximum regret ratio are in Sect. 2.1.

[8] presented the RMS problem: Given a dataset T, the family of all linear functions $\mathcal{L}$, and an integer K, compute a set $S \subseteq T$ of size K such that $mrr(S, L)$ is minimum. This problem has been proven NP-Complete in [5]. Hence, [8] and later work proposed heuristics. Nonetheless the computation is still time-consuming. [8] showed that it is sufficient to consider the Skyline set rather than the whole dataset as input to compute the regret minimization set. However the Skyline becomes with marginal benefit when its size grows, e.g., in anti-correlated setting. [6,7] proposed respectively the Top-K frequent skylines (Top-KF) and Top-K priority (Top-KP) skylines as candidates sets for computing the regret minimization set. They claimed that both operators speed up the RMS computation by up to two orders of magnitude. However, the empirical evaluation in that work are not conclusive. In this paper, we investigate the speed up provided by these two candidates sets. Concretely, we verify wrt several parameters (i) if these sets speed up RMS computation and (ii) impact the output regret. We consider the RMS state of the art algorithm *sphere* [12]. Moreover, we use NSC [1] an indexing structure to compute (i) Skyline, (ii) Top-KF and (iii) Top-KP sets.

2 Background

2.1 Regret Minimization Sets (RMS)

[8] presented the regret minimization queries RMS to avoid the limitations of skyline and Top-k queries. Unlike Top-K queries, RMS do not require scoring functions, and unlike Skyline queries, they bound the result size. The main idea is to select a subset S of a dataset T such that S minimizes the user *regret*. The *regret* represents how far the user's *best* tuple in S is from the user's *best* tuple in T. Specifically, reference [8] addressed the following problem:

> *Problem* ***RMS*** Given a dataset T, the family of all linear scoring function $\mathcal{L}$, an integer K, compute a set $S \subset T$ of size K that minimizes the maximum regret ratio $mrr(S, \mathcal{L})$.

In the following, we explain the maximum regret ratio of a set S wrt $\mathcal{L}$. Let $f \in \mathcal{L}$ be a scoring function. Given a dataset X, let $f_1(X)$ be the highest score by considering tuples in X. The regret of $S \subseteq T$ wrt a function f is $f_1(T) - f_1(S)$ and the regret ratio is $\frac{f_1(T)-f_1(S)}{f_1(T)}$. The maximum regret ratio is then $mrr(S, \mathcal{L}) = max_{f\in\mathcal{L}} \frac{f_1(T)-f_1(S)}{f_1(T)}$. [5] proved the NP hardness of RMS problem. The regret minimization set has been shown (i) scale-invariant, i.e., the maximum regret ratio remains the same even if the values in the dataset are multiplied by the same factor, and (ii) stable, i.e., the RMS does not change when *weak* tuples (tuples not having the highest score wrt any function) are inserted or deleted from the dataset. Sphere [12] is currently the state of the art heuristic algorithm. It has interesting time complexity and provides theoretical guarantees on the output. Its time complexity is $O(n \cdot e^{O(\sqrt{d \cdot ln(n)})} + n \cdot k^3 \cdot d)$ where n represents the dataset size, d the number of dimensions and k the output size. [8] showed that it suffices to consider the skyline set to compute the RMS rather than the whole dataset. In other words, the optimal solution S^* is only composed of skyline tuples. [9] presented an even smaller and accurate candidate set, namely *Happy* tuples. However, it is time-consuming. Its time complexity is $O(n^2 \cdot d^2)$.

2.2 Multidimensional Skyline

The multidimensional skyline or subspace skyline consists in considering subsets of the set of dimensions for skyline analysis. Given a set of dimensions D and a dataset T. Let $X \subset D$, $Sky(T, X)$ is the set of skyline points by considering only attributes in X. The Skycube [10] has been proposed to optimize the evaluation of the skyline wrt any subspace. It consists simply in materializing the results wrt any subspace. Since it requires an exponential space wrt the number of dimensions, [1,2,11] proposed summarization techniques. We note NSC [1] which stores in an intelligent way, for every tuple, the subspaces where the tuple is dominated. It has been shown as time and memory efficient.

The multidimensional skyline analysis of a dataset gives a useful insight on the best tuples within a dataset. For example, the frequency of a tuple is the number of subspaces in respect to which it belongs to its respective skyline. Let $t \in T$ $Frequency(t) = |\{X \subseteq D \ s.t. \ t \in Sky(T, X)\}|$. The tuple with the highest frequency may have the best values on the dimensions. Another interesting operator is called the skyline priority which simply is the size of the smallest subspace wrt to which the tuple belongs to its respective skyline. Let $t \in T$ $Priority(t) = min_{X \subseteq D | t \in Sky(X)}(|X|)$. A tuple with low priority may belong to several skylines.

In this paper, we want to investigate the impact of (i) Top-KF a ranking query based on the skyline frequency and (ii) Top-KP a ranking query based on the skyline priority on *sphere* performance, i.e., processing time and output regret. Given K, computing Top-KF and Top-KP requires exponential time wrt the numbers of dimensions d. Hence we use NSC for that purpose.

2.3 The NSC Structure

NSC (Negative SkyCube) stores for each tuple t a list of pairs, each summarizing the subspace where t is dominated. Let $t \in T$ and a let $p = \langle X|Y \rangle$ computed wrt some tuple $t' \in T$. X represents dimensions where t' is strictly better than t, and Y represents dimensions where t and t' are equal. p summarizes the set of subspaces where t' dominates t. This set, denoted by $cover(p)$ is equal to

$\{Z \subset D | Z \subseteq X \cup Y \textit{ and } Z \not\subseteq Y\}$. We give an example to illustrate NSC and we refer the interested reader to [1] for more details for this structure.

Example 2. Consider Table 4. We assume that small values are preferred for every dimension. The skyline of T wrt dimension A is $\{t_1, t_2\}$ because these two tuples have the least value of A. The skyline wrt dimensions AD is $\{t_1, t_2, t_3, t_4, t_5\}$. t_6 does not belong to the skyline because it is dominated by t_4: the later has a better value of A and a better value on D.

Table 4. Dataset T

Id	A	B	C	D
t_1	1	1	3	3
t_2	1	1	2	3
t_3	2	2	2	2
t_4	4	2	1	1
t_5	3	4	5	2
t_6	5	3	4	2

By comparing some tuple t to all the others, we obtain a set of *pairs* that summarizes the the subspaces where t is dominated. For example, comparing t_1 to t_2 returns the pair $\langle C|ABC\rangle$: t_2 is better than t_1 in C and these two tuples are equal on ABC. From this pair, we can deduce that, e.g., t_1 doesn't belong to the skyline wrt AC. Table 5 depicts the list of pairs associated to each tuple after comparing it to all the others. Note that pairs $\langle X|Y\rangle$ where $X = \emptyset$ are not stored because they do not bring any dominance information.

Table 5. List of pairs associated to every $t \in T$

Tuples	Pairs
t_1	$\langle C|ABD\rangle, \langle CD|\emptyset\rangle, \langle D|\emptyset\rangle$
t_2	$\langle D|C\rangle, \langle CD|\emptyset\rangle, \langle D|\emptyset\rangle$
t_3	$\langle AB|\emptyset\rangle, \langle AB|C\rangle, \langle CD|B\rangle$
t_4	$\langle AB|\emptyset\rangle, \langle A|B\rangle, \langle A|\emptyset\rangle$
t_5	$\langle ABC|\emptyset\rangle, \langle ABC|D\rangle, \langle BCD|\emptyset\rangle, \langle BC|D\rangle$
t_6	$\langle ABC|\emptyset\rangle, \langle ABC|D\rangle, \langle ABCD|\emptyset\rangle, \langle A|D\rangle$

In Table 5, some pairs can be seen as *redundant*. For example, pair $\langle D|\emptyset\rangle$ associated to t_1 tells that t_1 is dominated wrt D. The same information can be derived from $\langle CD|\emptyset\rangle$ that is associated to t_1 too. Hence, $\langle D|\emptyset\rangle$ without losing any information regarding dominance. The summarized sets of pairs are represented in Table 6. Note that the number of pairs decreases from 20 to 9. This is the NSC associated to the dataset T.

Table 6. NSC of Table T

Tuples	Pairs
t_1	$\langle C\|ABD\rangle, \langle CD\|\emptyset\rangle$
t_2	$\langle CD\|\emptyset\rangle$
t_3	$\langle AB\|C\rangle, \langle CD\|B\rangle$
t_4	$\langle AB\|\emptyset\rangle$
t_5	$\langle ABC\|D\rangle, \langle BCD\|\emptyset\rangle$
t_6	$\langle ABCD\|\emptyset\rangle$

Again, [1] gives all the details about how this summary is obtained, maintained in case of dynamic data and used to speed up skyline queries evaluation.

Algorithm 1 describes the procedure to compute Top-KF through NSC. We compute the subspaces where a tuple t is dominated by computing the *cover* of all pairs related to t (line 4–7). We then compute the score of each tuple and put them in list *Score* (line 8). We sort *Score* and select Top-K tuples (line 9–11). Algorithm for Top-KP is similar to Algortihm 1 with a difference in computing the score (line 8).

Algorithm 1: Top K frequent tuples

```
   Input: NSC, $T$, $K$, $D$
   Output: $Top - KF$
1  begin
2      $Top - KF \leftarrow \emptyset$
3      $Score \leftarrow []$
4      foreach $t \in T$ in parallel do
5          $E \leftarrow \emptyset$
6          foreach $p \in NSC[t]$ do
7              $E \leftarrow E \cup cover(p)$
8          $Score.append(t, 2^{|D|} - |E|)$
9      $sort(Score)$
10     foreach $i \in [0, K)$ do
11         $Top - KF \leftarrow Top - KF \cup Score[i].first$
12 return $Top - KF$
```

3 Experiments

In this section, we report on some of the experimental results we obtained so far. We focus our comments on three aspects:

1. We evaluate the speed up of RMS computation provided by considering the Skyline set as a candidate set.
2. We investigate the speed up and output regret of RMS algorithm *sphere* by considering Top-K Frequent and Top-K priority sets as candidates sets for a given K.
3. Given K, we evaluate the regret of Top-K frequent and priority sets.

Hardware & Software. we consider the state of the art algorithm *sphere* [12] for computing regret minimizing sets and the structure NSC [1] for computing (i) skyline, (ii) Top-K frequent and (iii) Top-K priority sets. All the experiments are conducted on a Linux machine equipped with two 2.6 ghz hexacore CPUs and 32GB RAM. Software is in C++ and available on GitHub[1].

Datasets. We consider synthetic datasets generated through the framework in [3]. The parameters considered for these experiments and their (default) values are illustrated in Table 7.

Table 7. Parameters

Parameters	Values
Distribution	Independent (INDE), Anti-correlated (ANTI)
n (dataset size)	$\mathbf{100K}, 1M$
d(number of dimensions)	$4, \mathbf{8}, 12$
k(output size)	$20, \mathbf{30}, 40, 60, 80, 100$

3.1 Speed up with Skyline Set

Here, we evaluate the speed up of *sphere* by considering the Skyline set (S(n)) as input instead of the whole data set (D(n)). Note that the output set and regret are the same whether we consider the skyline set or the whole dataset (Refer [8]). Figures 1 and 2 depict the results. It is also important to note that the reported execution time when the skyline is used as input data set includes the execution time used to obtain this skyline.

The first observation we can make is that using Skyline as input data set enables faster computation of the minimum regret set on all cases. That's, thanks to NSC structure, the skyline computation time is negligible compared to *Sphere*

[1] https://github.com/karimalami7/NSC.

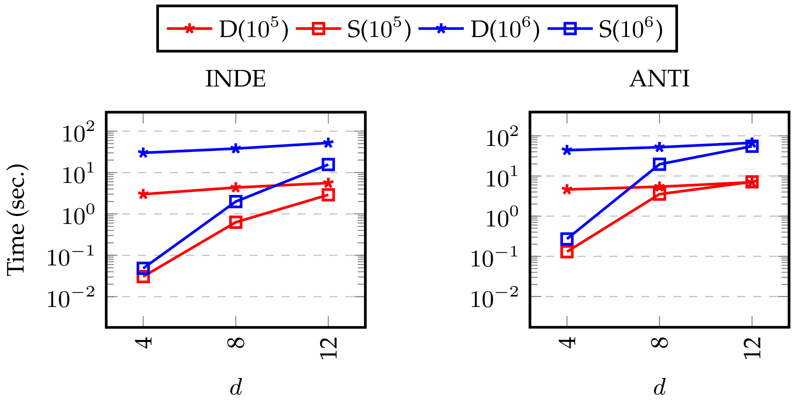

Fig. 1. Speedup of *Sphere* with Skyline as input set by varying dimensionality d

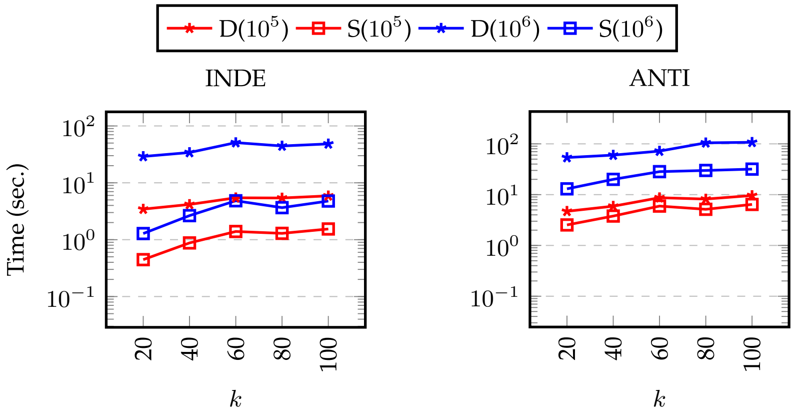

Fig. 2. Speedup of *Sphere* with Skyline as input set by varying the output size k

execution time. The second observation is that when d increases, the benefit of using the skyline as input decreases. This is because the skyline size gets closer to that of the entire data set. Hence, the benefit of *Sphere* gets smaller. For example, for a dataset with 1 million tuples with uniform distribution, the skyline set contains 418 tuples with 4 dimensions and 237726 tuples with 12 dimensions. This behavior is also observed when we compare INDE and ANTI distributions. With anti-correlated data, gain of using the skyline is already negligible when $d = 8$ with anti-correlated data which is not the case with independent uniform data distribution.

It is also interesting to note that the execution time of *Sphere* is almost constant wrt to K (the size of its output) (see Fig. 2.

We conclude that considering the Skyline set as candidate set has a limitation (its size is large when d is large), even if its computation time is negligible. In the next section, we investigate the impact of multidimensional skyline variations related ranking functions, i.e., Top-KF and Top-KP, on *sphere*.

3.2 Speedup and Regret of *Sphere* with Multidimensional Skyline

In this section, we evaluate the speedup of *Sphere* by providing Top-KF and Top-KP sets as input sets. Figure 3 depict the obtained results.

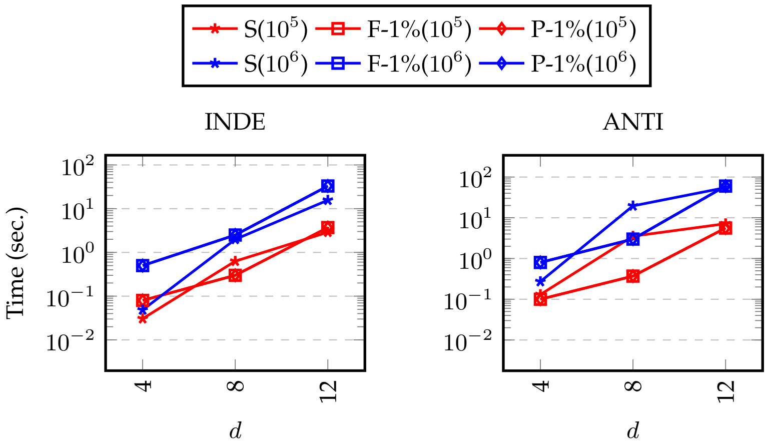

Fig. 3. Computation time of *sphere* with inputs sets (i) Skyline (ii) Top K Frequent and (iii) Top K Priority by varying d

Regarding computation time, we do not see big improvement by providing Top 1% tuples by frequency or priority. Indeed, *Sphere* computation time is improved because the input sets are smaller and have constant sizes (Top 1% instead of the whole skyline) however the computation time of input sets is now large and grows rapidly with increasing dimensionality. Indeed, computing Top-KF tuples requires the computation of an exponential number of skylines. Hence this computation time is not amortized by the fact that the obtained result is small.

3.3 Regret Ratio with Different Input Sets

In this section we analyze the quality of *Sphere* output when using different input data sets by contrast to the previous section where we analyzed just the execution time. Figures 4 and 5 show the results obtained from the same data sets as those used in the previous section. We see that all input sets provide similar regret ratio. We also see that for small k (under 60) when considering Top 1% frequent tuples as input sets, the regret ratio computed by *Sphere* is better than that of the output when the skyline is used as input set. This is explained by the fact that *sphere* is actually a heuristic approach to compute RMS. Indeed, Top 1% frequent tuples discard some *noisy* points that are consequently not selected by *Sphere*.

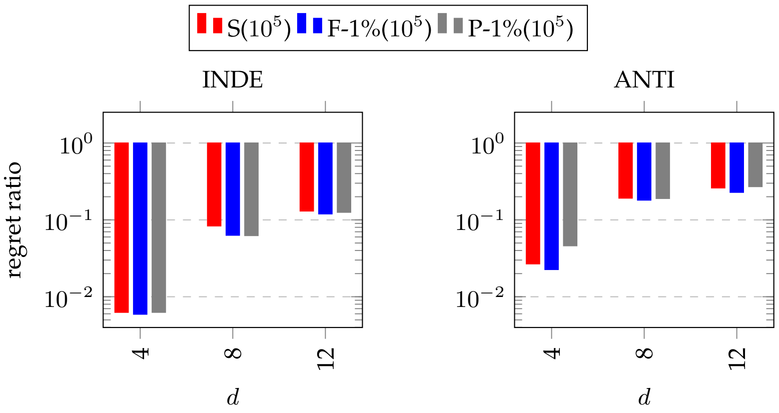

Fig. 4. Regret of *sphere* by input sets (i)Sky (ii) TopKF (iii) TopKP and varying d

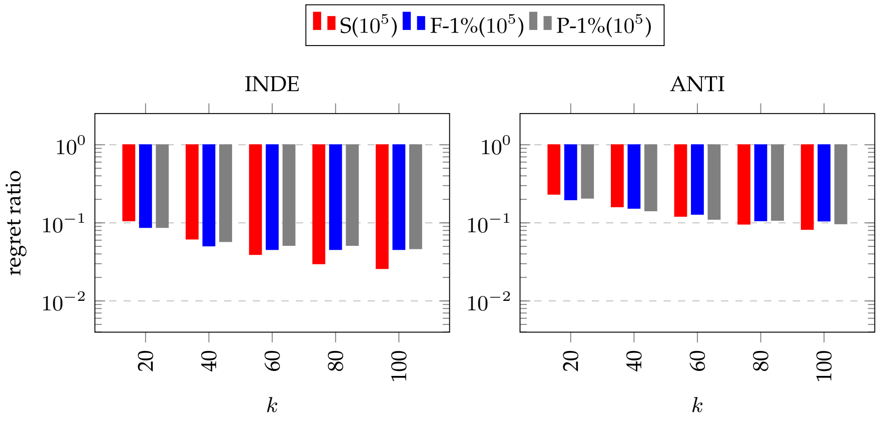

Fig. 5. Regret of *sphere* by input sets (i)Sky (ii) TopKF (iii) TopKP and varying k

3.4 Multidimensional Skyline Metrics as Alternatives to RMS Algorithms

So far, we showed that Top-KF and Top-KP provide good input sets for *sphere*. Now we want to answer the question: Can Top-KF or Top-KP (without *sphere*) compute sets that achieve regret ratio close to that achieved by *sphere*? Concretely, we evaluate the regret ratios of sets of size K computed with (i)*sphere* (ii) TopKF and (ii) TopKP. Figures 6 and 7 depict the results. Globally, we can see that TopKF achieves a good regret ratio when dimensionality gets large and k is small. One possible explanation of this behavior would be the fact that higher dimensionality means a higher number of skylines. This makes tuples better differentiated. Indeed, with lower dimensionality, many tuples share the same score(number of skyline they belong to) which makes them hardly distinguishable.

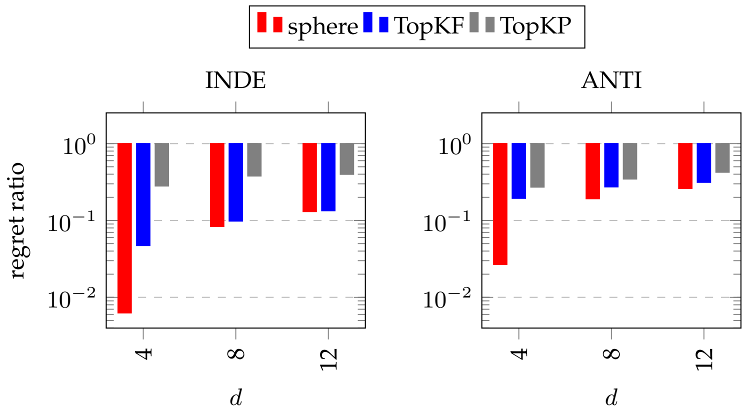

Fig. 6. Regret of (i) *sphere* (ii) TopKF (iii) TopKP by varying k

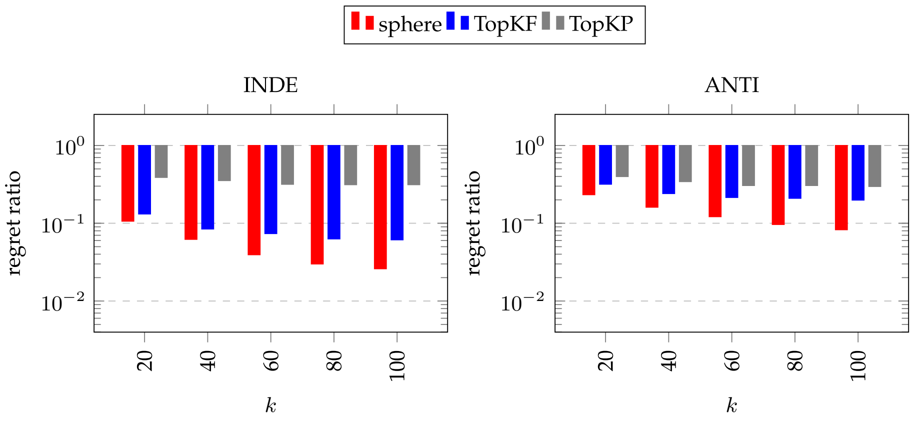

Fig. 7. Regret of (i) *sphere* (ii) TopKF (iii) TopKP by varying k

4 Discussion

The experimental results we obtained show that NSC is useful for RMS algorithms as it computes very efficiently different variants of input data sets that are provided to RMS algorithms such as *Sphere*. Some of these input data sets can be even used as *an approximations* of RMS results. Indeed, our empirical results showed that some of them already provide small regrets. From the practical point of view, this is very important since the *Sphere* application on even small input data sets can be prohibitive when the number of dimensions is large. The encouraging empirical results obtained so far should be pursued to state some theoretical error guarantee bounds wrt chosen input data sets. This is our plan for future research.

References

1. Alami, K., Hanusse, N., Wanko, P.K., Maabout, S.: The negative skycube. Inf. Syst. **88**, 101443 (2020)
2. Bøgh, K.S., Chester, S., Sidlauskas, D., Assent, I.: Template skycube algorithms for heterogeneous parallelism on multicore and GPU architectures. In: Proceedings of SIGMOD Conference, pp. 447–462 (2017)
3. Börzsönyi, S., Kossmann, D., Stocker, K.: The skyline operator. In: Proceedings of ICDE Conference, pp. 421–430 (2001)
4. Chaudhuri, S., Gravano, L.: Evaluating top-k selection queries. VLDB **99**, 397–410 (1999)
5. Chester, S., Thomo, A., Venkatesh, S., Whitesides, S.: Computing k-regret minimizing sets. Proc. VLDB Endowment **7**(5), 389–400 (2014)
6. Han, S., Zheng, J., Dong, Q.: Efficient processing of k-regret queries via skyline frequency. In: Meng, X., Li, R., Wang, K., Niu, B., Wang, X., Zhao, G. (eds.) WISA 2018. LNCS, vol. 11242, pp. 434–441. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-02934-0_40
7. Han, S., Zheng, J., Dong, Q.: Efficient processing of k-regret queries via skyline priority. In: Meng, X., Li, R., Wang, K., Niu, B., Wang, X., Zhao, G. (eds.) WISA 2018. LNCS, vol. 11242, pp. 413–420. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-02934-0_38
8. Nanongkai, D., Sarma, A.D., Lall, A., Lipton, R.J., Xu, J.: Regret-minimizing representative databases. Proc. VLDB Endowment **3**(1–2), 1114–1124 (2010)
9. Peng, P., Wong, R.C.: Geometry approach for k-regret query. In: Cruz, I.F., Ferrari, E., Tao, Y., Bertino, E., Trajcevski, G. (eds.) IEEE 30th International Conference on Data Engineering, Chicago, ICDE 2014, IL, USA, March 31 - April 4, 2014, pp. 772–783. IEEE Computer Society (2014)
10. Tao, Y., Xiao, X., Pei, J.: Subsky: efficient computation of skylines in subspaces. In: 22nd International Conference on Data Engineering (ICDE 2006), pp. 65–65. IEEE (2006)
11. Xia, T., Zhang, D., Fang, Z., Chen, C.X., Wang, J.: Online subspace skyline query processing using the compressed Skycube. ACM TODS **37**(2), 15:1–15:36 (2012)
12. Xie, M., Wong, R.C., Li, J., Long, C., Lall, A.: Efficient k-regret query algorithm with restriction-free bound for any dimensionality. In: Das, G., Jermaine, C.M., Bernstein, P.A. (eds.) Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, 10–15 June 2018, pp. 959–974. ACM (2018)