Distrib Parallel Databases (2016) 34:535–566
DOI 10.1007/s10619-015-7187-9

Sliding window top-k dominating query processing over distributed data streams

Daichi Amagata[1] · **Takahiro Hara**[1] · **Shojiro Nishio**[1]

Published online: 10 October 2015

Abstract Preference query processing is important for a wide range of applications involving distributed databases, such as network monitoring, web-based systems, and market analysis. In such applications, data objects are generated frequently and massively, which presents an important and challenging problem of continuous query processing over distributed data stream environments. A top-k dominating query, which has been receiving much research attention recently, returns the *k* data objects that dominate the highest number of data objects in a given dataset, and due to its dominance-based ranking function, we can easily obtain *superior* data objects. An emerging requirement in distributed stream environments is an efficient technique for continuously monitoring top-k dominating data objects. Despite of this fact, no study has addressed this problem. In this paper, therefore, we address the problem of continuous top-k dominating query processing over distributed data stream environments. We present two algorithms that monitor the exact top-k dominating data and efficiently eliminate unqualified data objects for the result, which reduces both communication and computation costs. In addition to these algorithms, we present an approximate algorithm that further reduces both communication and computation costs. Extensive experiments on both synthetic and real data have demonstrated the efficiency and scalability of our algorithms.

✉ Daichi Amagata
amagata.daichi@ist.osaka-u.ac.jp

Takahiro Hara
hara@ist.osaka-u.ac.jp

Shojiro Nishio
nishio@ist.osaka-u.ac.jp

[1] Department of Multimedia Engineering, Graduate School of Information Science and Technology, Osaka University, 1-5 Yamadaoka, Suita, Osaka 565-0871, Japan

Keywords Top-k dominating queries · Multi-dimensional data · Distributed data streams

1 Introduction

Recently, preference queries, which retrieve only preferable data objects from a multi-dimensional dataset, have been receiving significant research attention in the database community. These types of queries provide a wide range of multi-criteria decision making applications with much benefits, for example, multimedia retrieval [32], web search [29], market analysis [38], and e-commerce [38]. Two of the most widely used preference queries are top-k and skyline queries.

Given a ranking (scoring) function $f : \mathbb{R}^d \rightarrow \mathbb{R}$, where d is the number of attributes (dimensionality), a top-k query [6] retrieves the k data objects with the best score. The great advantage of top-k queries is that users can control the number of data objects in the result set by specifying the parameter k. The drawback of the queries, on the other hand, is that it is not always easy for users to specify an appropriate ranking function [10].

Skyline queries [3] overcome this drawback because this query does not require any ranking functions. The result of a skyline query consists of data objects not dominated by other data objects in a given dataset. The dominance relationship is defined as: given two data o_i and o_j, o_i dominates o_j if o_i is not worse than o_j in all attributes and better than o_j in at least one attribute. Through skyline query processing, users can obtain data objects that are not worse than others. However, users cannot control the size of the result set, which may overwhelm users.

Top-k dominating queries [24,36,37] are powerful queries that combine the advantages of both top-k and skyline queries. More specifically, the top-k dominating queries require no ranking functions of users and can control the size of the result set. Top-k dominating queries retrieve the k data objects that dominate the highest number of data objects in a given dataset. That is, the score of a data object o is the number of data objects that o dominates. Top-k dominating queries identify the most important data objects in an intuitive way. This can be helpful for many applications including the above examples.

1.1 Motivation and challenges

With the recent advance of large-scale data centers and cloud computing, data generation and storage are becoming increasingly massive and distributed [11]. In such environments, data objects are frequently generated and updated at distributed data sites, and centralizing data objects is not feasible [2,7,8,25,26]. Therefore, we need a distributed computing environment in the case where data sources (sites) are distributed, and many applications require *distributed continuous query processing* to monitor users-preferable data objects. We consider the following application examples that can obtain benefits by employing distributed continuous top-k dominating queries.

Example 1 Consider a stock market monitoring system consisting of a single central server and some distributed data sites, e.g., NYSE, NASDAQ, and so on. Each data site receives a local data stream and the central server plays a role of reporting the user-requiring stocks obtained from data sites [21,30]. In continuous top-k dominating query processing, each stock data o is given its score, which is the number of stocks that o dominates, e.g., in terms of price and volume. We can track k stocks with worth investing (i.e., the k-highest score), without priori knowledge (i.e., specifying scoring functions), among the distributed database consisting of the server and distributed data sites.

Example 1 highlights the effectiveness of top-k dominating queries since users can see intuitively good stocks while controlling the result size. In the same view of Example 1, an online auction system consisting of several databases can also use a distributed continuous top-k dominating query, and the system always displays the k data (products) as recommendation items. IP-network monitoring is also an application of distributed continuous top-k dominating query processing because its data sources are inherently distributed, e.g., one central server monitors traffic streams from local data sites.

Example 2 Consider a network monitoring system consisting of a single central server and some data sites observing local data streams [2,7,8,26]. Dominance-based approaches have been widely accepted in network monitoring environments [21,25,41], but skyline queries cannot control the result size, which may make it difficult for administrators to follow the results. (Attributes of data objects in this example can be traffic and the number of successful transactions.) A continuous top-k dominating query, which is also dominance-based query, would be useful, because through distributed continuous top-k dominating query processing, administrators can analyze network troubles by monitoring a set of reasonable sized (i.e., k) data objects.

Other approaches have also been considered for network monitoring, for example, point queries and heavy hitter [12]. However, they require a specific parameter, e.g., a data point in point queries and ϕ that determines frequency of data appearance in heavy hitters. To specify such parameters, users need previous knowledge, while top-k dominating queries require only k as input.

We see that many modern applications including above ones rely on the distributed data streaming model, and need efficient continuous query processing techniques. Such techniques should satisfy computation-efficiency and also communication-efficiency. This is because an important requirement in distributed stream environments is small communication cost [2,7,8,21,25,26,41].

We here claim that the existing techniques for top-k dominating query processing cannot solve our problem efficiently. The proposed techniques in [9,34,36,37,40] focus on static data and *one-time* query processing. Continuous top-k dominating query processing techniques in [14,15,27] assume a centralized setting, which is infeasible in distributed environments. An emerging and important challenge is developing an efficient technique for continuous top-k dominating query processing in distributed data stream environments. This problem poses more challenges over centralized settings, because we can obtain no global information without retrieving *all* data objects, but if we do so, too much communication cost is incurred. Thus the first challenge is to design a scheme that data sites can judge whether newly generated data objects are possible to be the top-k dominating data or not. This scheme avoids forwarding unqualified data objects for the top-k dominating data. The second challenge is to eliminate unnecessary score computation. Note that it is necessary for a data object o to check its dominance relationships with all others to obtain its exact score. Since data objects are distributed over data sites, we cannot compute the exact scores of data objects locally. This means that we should compute the scores by a distributed approach. Due to the dominating-count based scoring function, the scores of data objects may vary in dynamic environments where data generations and expirations occur. We have to reduce the number of data objects to compute their exact scores, because the distributed approach incurs communication cost. We address these challenges in this paper.

For continuous data monitoring, we adopt a time-based sliding window model [1], where the data objects generated within W time-stamps from the current time-stamp are target for monitoring. In this paper, we propose two approaches for *exact* top-k dominating data monitoring. One is a filter-based approach that employs data objects with high-dominance power as filters to avoid unnecessary data forwarding. The other is a cache-based approach that aims at further reduction of unnecessary data forwarding. In both approaches, we present a technique for distributed top-k dominating data computation. To reduce both communication and computation costs, we should avoid re-calculation of the top-k dominating data as much as possible. To this end, we propose a lower- and upper-bounding technique for scores of given data objects. By exploiting this technique, we carefully select candidates of the result set from the entire dataset. That is, our approaches are designed so that the query result can be incrementally updated. We also propose a sampling-based *approximate* approach for the top-k dominating data computation that further reduces both communication and computation costs while keeping high accuracy.

1.2 Contributions and organization

We summarize our contributions as follows.

- We investigate the problem of processing continuous top-k dominating queries in distributed data stream environments. To the best of our knowledge, this problem has not yet been addressed.
- We propose two efficient algorithms for distributed continuous top-k dominating query processing. Both of them significantly reduce communication and computation costs.
- We also propose an approximate algorithm for the top-k dominating data computation. This algorithm further reduces both communication and computation costs compared with our exact algorithms.
- Extensive experiments on both synthetic and real data demonstrate the efficiency and scalability of our algorithms.

The rest of this paper is organized as follows. Section 2 reviews related work. Section 3 provides the preliminaries to present our algorithms, introduces our assumption, and formally defines the problem. We describe the filter-based algorithm in Sect. 4, the cache-based algorithm in Sect. 5, and the sampling-based approximate algorithm in Sect. 6, respectively. We show the results of our experiments in Sect. 7. Finally, in Sect. 8, we conclude the paper.

2 Related work

Along with preference queries, other continuous queries in distributed data streams [7,8,26] have been developed recently. However, these works study techniques for processing of queries with essentially different semantics (e.g., join and threshold-monitoring). So, in this section, we focus on the existing continuous top-k and skyline query processing techniques and review them. This is because top-k dominating queries employ a special ranking function instead of user-specified ranking functions in top-k queries, and this ranking function is based on the dominance relationship utilized in skyline queries. In the end of this section, we introduce prior works of top-k dominating query processing in centralized databases.

2.1 Top-k queries

Efficient top-k query processing techniques in data stream environments have been studied in the context of both centralized [6,23,35,38] and distributed databases [2]. In [6], query processing is performed by *snapshot* approach. The algorithm proposed in [38] focus on processing a lot of top-k queries and solve by a geometric approach. In [23,35], top-k monitoring is performed with a sliding window. The algorithm proposed in [23] reduces the size of the candidate set for top-k data by taking into account a pair of score and expiration time. In [35], the candidate set is minimized and its computation time becomes faster than that of [23]. Continuous top-k query processing is performed in distributed databases in [2]. However, in [2], data is represented by one-dimension, that is, quite different from our problem. Some studies, e.g., [33], focus on multi-dimensional top-k query processing in distributed environments, but their techniques are only for static data, meaning that it is non-trivial to apply such techniques for streaming data.

2.2 Skyline queries

As well as top-k queries, continuous skyline queries [16] have also been studied well for moving objects [5,13] and sliding window model [18,22,31]. All of these studies, however, deal with skyline queries at a centralized database, and they are hard or inefficient to adapt to distributed settings.

Techniques for both snapshot and continuous skyline query processing in distributed environments have been well developed [11]. In [41], the authors assume a client-server model, and monitor *frequent skyline* which is the set of data objects existing as skyline points within a user-specified interval time. They reduce communication cost by exploiting minimum bounding rectangle (MBR) based filters. In [20,21], distributed continuous skyline query processing problem has been solved as a case study to reduce both communication and computation costs. The study of [30] also assumes sliding window based skyline monitoring in distributed environments. The proposed technique in this study utilizes similar idea with [23] and prunes data points that are dominated by others until their expiration time. Recently, efficient monitoring approach for *fragmented skyline* has been proposed [25]. This study assumes fragmented data over data sites, i.e., accurate value on a given dimension of a data object is not known at a data site, which is different from our assumption. Although the above methods can efficiently monitor skyline, they cannot control the size of the result set as mentioned in Sect. 1. To overcome this problem, representative skyline queries [4,19], which select k skyline objects from skyline, have been proposed. However, top-k dominating queries retrieve data objects *not only from skyline*, thereby the problem in this paper differs from skyline queries.

It is important to note that the problem in this paper is much more complex than the continuous skyline problem. Monitoring skyline needs to care changes of dominance relationships between current skyline data and newly arriving data. Continuous top-k dominating queries further needs to care ranking of data objects. Because scores of data objects are based on dominating count, data generations and expirations lead to changes of the scores, meaning that ranks of data objects may also vary. To avoid large communication and computation costs, we have to tight the size of candidate set. We solve this non-trivial problem in this paper.

2.3 Top-k dominating queries

Top-k dominating queries have first been proposed in [24], then, in [36,37], efficient algorithms for retrieving the top-k dominating data objects with aR-tree have been proposed. Some alternatives have also been proposed. In [17,39], the authors have studied an efficient algorithm for top-k dominating queries in uncertain databases. Top-k dominating queries are also processed under the assumption that data objects are vertically decomposed [9,40]. A study in [34] assumes spatial data objects, and proposed top-k neighborhood dominating queries, which take into account spatial attributes. Top-k dominating queries in metric space have also been proposed [32]. Continuous top-k dominating queries have been studied in [14,15,27]. Taking into account the expiration time of the top-k data, a scheduling-based approach has been proposed in [15]. A graph-based indexing approach enhances continuous top-k dominating query processing, resulting in faster computation time.

Although the above techniques are efficient in a centralized database, these require global information, e.g., to build the graph and aR-tree, and incurs significant communication cost in distributed environments. Also, the studies of continuous top-k dominating queries in a centralized database basically assume a count-based sliding window where one data generation leads to one data expiration. This assumption enables update-procedures to be low overhead because of only a single data generation and expiration. However, it is obvious that this assumption cannot deal with high-speed data streams which typically appear in distributed data streaming environments [7,26]. We can therefore see that applying the existing centralized algorithms to distributed databases is clearly non-trivial and inefficient. An efficient technique for continuously monitoring top-k dominating data is an emerging issue in distributed data stream environments. In Sect. 7, we show that the centralized approach is inefficient in a distributed environment where many data objects are generated at a time-stamp, but our approaches function well even in the environment.

3 Preliminaries

In this section, we first introduce our assumption of data and system models, then define the problem of this paper. Table 1 summarizes the symbols used in this paper.

3.1 Assumption

We assume a distributed database consisting of a single central server H and m distributed data sites $S_1, S_2, \ldots, S_m$, where S_i can directly communicate with H, which is a typical distributed setting and has been often applied in the past literatures [2,20,25,30]. Each data site S_i maintains its own d-dimensional dataset O_i, and the central server H also maintains its own d-dimensional dataset O_H that holds data objects forwarded from data sites. Let O be a global dataset, i.e., $O = O_1 \cup O_2 \cup \cdots \cup O_m$. A data object $o_i \in O$ can be represented as a point p_i, and $p_i = \{p_i[1], p_i[2], \ldots, p_i[d]\}$, where $p_i[j]$ is an attribute value of the jth dimension. We use data *objects* and data *points* interchangeably in this paper, and without loss of generality, we assume that smaller values are preferable. We assume a distributed stream environment, where each data site may generate some data points every time-stamp. The arrival and the expiration time of a data point p_i is denoted as p_i.arr and p_i.exp, respectively. Many applications, e.g., network monitoring, stock market monitoring, and e-commerce, may want to continuously monitor the top-k dominating data from the most recent data. Therefore, we employ a time-based sliding window of W length, which is more suitable for distributed environments than a count-based sliding window. This is because each data site independently generates data points and the number of newly generated data may not be constant. Our algorithm, however, can

Table 1 Overview of symbols

Symbol	Description
H	The central server
m	The number of data sites
S_i	The ith data site
O	The global d-dimensional dataset
O_i (O_H)	The d-dimensional database of S_i (H)
p_i	The ith data point
W	The length of the sliding window
t_{cur}	The current time-stamp
TOPK	The set of top-k dominating data points
CAND	The set of candidate of *TOPK*

be easily extended to the case of count-based sliding window. Let t_{cur} be the current time-stamp, and if a data point p_i is generated at t_{cur}, i.e., $p_i.\text{arr} = t_{cur}$, p_i will expire at $t_{cur} + W$, i.e., $p_i.\text{exp} = t_{cur} + W$ in the time-based sliding window. We say that p_i is active if p_i is in the window, that is, has not expired yet. At $t_{cur} = 0$, each data site starts data generation, and our algorithms starts working from $t_{cur} = W - 1$, i.e., when the sliding window size is satisfied.

3.2 Problem definition

Top-k dominating queries give a data point p_i to its score, $score(p_i)$, through checking dominance relationships with other data points. Therefore, we first define the concept of *dominance* below.

Definition 1 (Dominance). p_i is said to *dominate* $p_{i'}$ (denoted as $p_i \prec p_{i'}$) if they satisfy the following condition.

$$(\forall j \in [1, d], p_i[j] \leq p_{i'}[j]) \wedge (\exists j \in [1, d], p_i[j] < p_{i'}[j]) \tag{1}$$

Example 3 Figure 1 illustrates a two-dimensional dataset of ten data points, and p_3 dominates p_2 because all the attribute values of p_3 are less than those of p_2.

We utilize a multi-dimensional grid like that shown in Fig. 1, and each data site and H maintain this data index structure for fast dominance computation. Each grid cell has its own identifier and maintains both *fully* and *partially* dominating cells.

Example 4 In Fig. 1, c_{11} maintains $\{c_{22}, c_{23}, c_{32}, c_{33}\}$ as fully dominating cells, because any data points in c_{11} dominate all the data points in those cells. c_{11} also maintains $\{c_{12}, c_{13}, c_{21}, c_{31}\}$ as partially dominating cells, because points in c_{11} may dominate points in those cells. In terms of the data points in c_{11}, data points in neither fully nor partially dominating cells are not dominated by the data points in c_{11}, thus

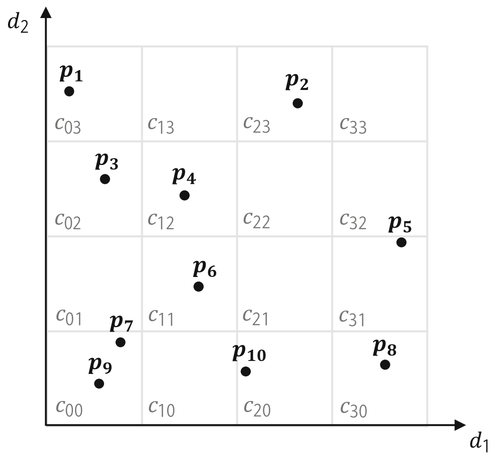


Fig. 1 A two-dimensional dataset and grid cells

the data points in c_{11} only have to check dominance relationship with data points in partially dominating cells. This accelerates dominance computation.

Newly generated or obtained data points are assigned to a specific cell, and the overhead of this operation is trivial because no high maintenance cost is incurred, as in MBR-based index structures. Note that, however, our algorithms do not depend on the multi-dimensional grid, and other index approaches can be adaptable, if necessary.

For a data point p_i, the number of data points dominated by p_i shows how superior p_i is in a given dataset. A top-k dominating query retrieves the k best data points in terms of this concept. The definition of continuous top-k dominating queries is as follows.

Definition 2 (Continuous top-k dominating queries). Consider an active data point $p_i \in O$, and the $score(p_i)$ is calculated as follows.

$$score(p_i) = |\{p_j \in O \mid p_i \prec p_j\}| \tag{2}$$

Given a set of active data points, a continuous top-k dominating query monitors data points with the k highest scores, TOPK.

Example 5 In Fig. 1, we assume that ten data points are active at the current time-stamp. In this case, if a user sets 2 as k, $TOPK = \{p_7, p_9\}$, because their scores are 4 and 8 respectively and larger than those of the others. We further assume that p_9 expires at the next time-stamp, meaning that *TOPK* will vary. A continuous top-k dominating query always monitors *TOPK*, so we have to retrieve the updated top-2 data points at the next time-stamp.

To achieve efficient top-k dominating data monitoring in distributed data stream environments, it is necessary not to consume network bandwidth wastefully, because communication-efficiency is particularly important for monitoring in distributed environments [2,7,8,26]. In addition, real time monitoring (i.e., computation-efficiency) is required for many applications. Our main objective is, therefore, to minimize total communication cost for top-k dominating data monitoring and also minimize top-k dominating data computation time.

3.3 Naive algorithm

A naive solution for the problem in this paper is that each data site forwards all newly generated data points to H, then H locally computes the scores of active data points and obtains the top-k dominating data. Repeating this operation at every time-stamp, we can correctly monitor the top-k dominating data. However, this centralized solution is obviously inefficient in terms of both communication and computation costs, and hard to achieve our objective. This motivates us to develop a sophisticated solution enabling efficient processing of continuous top-k dominating queries over distributed databases. Therefore, we propose distributed algorithms that can efficiently monitor the top-k dominating data. This is the main contribution of this paper.

4 Filter-based solution

In this section, we introduce our strategy for eliminating data points that never be the top-k dominating data, and then present our algorithm employing filter points to reduce communication cost. To this end, we define an important concept of *d-score* (dominated-score) as follows.

Definition 3 (d-score). Given an active data point $p_i \in O$, the d-score of p_i, $\phi(p_i)$, is calculated as follows.

$$\phi(p_i) = |\{p_j \in O \mid p_j \prec p_i\}| \tag{3}$$

Example 6 In Fig. 1, $\phi(p_2)$ is 6 because p_2 is dominated by $\{p_3, p_4, p_6, p_7, p_9, p_{10}\}$.

Based on the above definitions, we can show a property of top-k dominating queries.

Property 1 Given a set of active data points, data points with d-scores larger than $k-1$ cannot be top-k dominating data.

Due to Property 1, we can see that H needs only data points with d-scores less than k, which can reduce the number of candidate points for the top-k dominating data from the entire set of active data points. Interestingly, such a set of data points is k-skyband [24], so we formally define this dataset and describe an important property as follows.

Definition 4 (k-skyband). k-skyband is a set of active data points with d-scores less than k.

Property 2 TOPK belongs to k-skyband [24].

In fact, we can convert the top-k dominating data monitoring problem into two problems of k-skyband monitoring and top-k dominating data computation. Note that although past studies, e.g., [23], utilize k-skyband for continuous top-k monitoring, our work differs from them. Since we have to maintain k-skyband with small communication cost, a different and distributed approach is necessary to achieve that. Because of dynamic data environment, the scores of data points also dynamically change, so if some data points included in k-skyband are not forwarded to H, *TOPK* may not be correct. To monitor *TOPK* at H, therefore, we have to continuously maintain k-skyband efficiently and carefully select data points with non-zero probability of being the top-k dominating data at t_{cur}. Our algorithm solves these problems efficiently.

4.1 Overview

In the filter-based algorithm, we maintain k-skyband with small communication cost. To achieve this, we introduce n filter points that avoid unnecessary data forwarding from data sites. We know, from Property 2, that broadcasting k-skyband at every timestamp leads to monitoring the exact scores of data points, i.e., the correct *TOPK*. However this approach may incur large communication cost, since the size of k-skyband is larger than k. We show that we can reduce the size of candidate of *TOPK*, namely *CAND*, by computing the lower- and upper-bound scores of data points in k-skyband. Filter-based algorithm is designed to exploit these approaches, and updated *TOPK* incrementally and efficiently.

In Sect. 4.2, we first present the distributed manner for top-k dominating data computation, and then how to select *CAND* in detail. In Sect. 4.3, we present how to select n filter points and how to set n. We show the filter-based algorithm in Sect. 4.4.

4.2 Top-k dominating data computation

4.2.1 Distributed score computation

From Property 2, we can see that H obtains the top-k dominating data at t_{cur} through computing the scores of data points in the k-skyband. The k-skyband is a small subset of the entire dataset. This means that H does not hold all active data points and cannot compute the exact scores of data points locally. We therefore need a distributed approach to compute the scores of data points efficiently. The following equation gives a key to achieving this.

$$score(p_i) = \sum_{i'=1}^{m} |\{p_j \in O_{i'} \mid p_i \prec p_j\}| \tag{4}$$

This equation is a simple transformation of Equation (2). Given an active data point p_i, let $score(p_i, O)$ and $score(p_i, O_j)$ be its global score, i.e., $score(p_i) = score(p_i, O)$ and its local score, respectively. From Equation (4), $score(p_i, O)$ is computed by summing up local scores. To obtain scores of data points, therefore, H broadcasts the set of data points to all data sites. Each data site S_j computes local scores of the data points, and then sends $score(p_i, O_j)$ of all broadcasted data points to H. H computes the global scores of the data points through Equation (4), and can obtain the top-k dominating data. The advantage of this distributed approach is that each data site computes the scores of broadcasted data points in parallel, so the top-k dominating data computation time becomes faster than the centralized approach. At $t_{cur} = W - 1$, each site S_j calculates its local k-skyband of O_j, by computing d-scores of data points with utilizing the grid, and forwards it to H. H calculates the global k-skyband from the received local k-skyband sets, then performs the distributed score computation. As a result, H obtains the correct *TOPK*. This distributed score computation approach is denoted by DIST- SCORE- COMP$(\cdot)$, where its input is the set of data points whose scores are computed.

4.2.2 Candidate selection

As explained, H has the exact scores of data points at the time-stamp of $W - 1$, since we compute the scores of all data points in k-skyband. We here explain how to update *TOPK* incrementally. When $t_{cur} \geq W$, calculating the maximum possible increment of scores (upper-bound) and the minimum possible decrement of scores (lower-bound) provides us with the set of data points which never be the top-k dominating data at t_{cur}.

Let n_g and n_e be the number of newly generated and expired data at t_{cur}, respectively. The following lemma eliminates the data points that never be the top-k dominating data at t_{cur}.

Lemma 1 *Let $score_k$ be the kth highest score at the previous time-stamp. Its lower-bound score at t_{cur} is $score_k - n_e$. Given an active data point $p_i \notin TOPK$, let $score(p_i)^{t_{cur}-1}$ be the score of p_i at the previous time-stamp. If $score(p_i)^{t_{cur}-1} + n_g$, i.e., upper-bound score of p_i at t_{cur}, is less than $score_k - n_e$, p_i cannot be the top-k dominating data at t_{cur}.*

Proof Let $score_j$ be the $j(\leq k)$th highest score at the previous time-stamp, and we have $score_1 - n_e \geq score_2 - n_e \geq \cdots \geq score_k - n_e$. If $score(p_i)^{t_{cur}-1} + n_g$ is less than $score_k - n_e$, there is at least k data points with larger scores than $score(p_i)$ at t_{cur}. Therefore, we can conclude that Lemma 1 is true. □

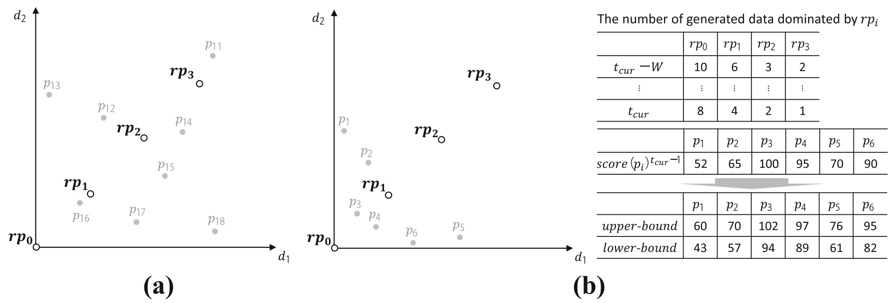
Based on Lemma 1, H can select the data points with non-zero probability to be the top-k dominating data from k-skyband, and the set of such data points is *CAND*. If $|CAND| = k$, *TOPK* does not vary, thus H does not need to broadcast *CAND*, meaning that no communication and computation costs are incurred for the top-k dominating data computation. Otherwise, H broadcasts *CAND* and performs DIST-SCORE-COMP(*CAND*). $\forall p_i \in O_H \setminus CAND$ cannot obtain its exact score, although the score is definitely less than the kth highest score. Hence $score(p_i)$ is set as $score(p_i)^{t_{cur}-1} + n_g$.

The above approach can select *CAND* and reduce both communication and computation costs because $|CAND|$ is typically smaller than the size of k-skyband. The approach however considers only n_g and n_e, so upper-bound and lower-bound scores are likely to be far from the actual score. Moreover, the approach is not scalable to the environments where a lot of data points are generated at every time-stamp. In such cases, if an active data point p_i is not included in *CAND* at t_{cur}, p_i tends to be included in *CAND* at the next time-stamp because its upper-bound score can be very high. To overcome this problem, we have to compute upper-bound and lower-bound scores tightly. We therefore introduce *reference points* to realize this idea.

Definition 5 (Reference points). Given an interval length of l, a reference point rp_i is $rp_i = \{l \cdot i, l \cdot i, \ldots, l \cdot i\}$, where $0 \leq i \leq r - 1$, and r is the number of reference points.

Note that l and r are system parameters. Figure 2a illustrates an example of a set of four reference points. Assume that in Fig. 2a, $\{p_{11}, \ldots, p_{18}\}$ is a set of data points generated at t_{cur}. For example, rp_1 dominates four newly generated data points as shown in Fig. 2b. The set of these reference points brings tight upper-bound and lower-bound scores with reasonable cost because given an active data point p_i, H can see the *certain* number of generated (or expired) data points dominated by p_i and the *possible* number of generated (or expired) data points dominated by p_i. More specifically, let $n_{rp_j}^{t_{cur}}$ be the number of data points generated at t_{cur} and dominated by rp_j. The upper-bound score of p_i is

$$score(p_i)^{t_{cur}-1} + n_{rp_j}^{t_{cur}} - n_{rp_{j'}}^{t_{cur}-W}, \tag{5}$$



The number of generated data dominated by rp_i

	rp_0	rp_1	rp_2	rp_3
$t_{cur}-W$	10	6	3	2
⋮	⋮	⋮	⋮	⋮
t_{cur}	8	4	2	1

	p_1	p_2	p_3	p_4	p_5	p_6
$score(p_i)^{t_{cur}-1}$	52	65	100	95	70	90

	p_1	p_2	p_3	p_4	p_5	p_6
upper-bound	60	70	102	97	76	95
lower-bound	43	57	94	89	61	82

Fig. 2 Our strategy for *CAND* selection through tightly computing upper-bound and lower-bound scores. **a** A set of two-dimensional reference points. **b** A list of the number of data points dominated by rp_i and upper-bound and lower-bound computation

where rp_j is the nearest neighbor point that dominates p_i and $rp_{j'}$ is the nearest neighbor point that is dominated by p_i. Also, the lower-bound score of p_i is

$$score(p_i)^{t_{cur}-1} + n_{rp_{j'}}^{t_{cur}} - n_{rp_j}^{t_{cur}-W}. \tag{6}$$

Example 7 In Fig. 2b, $score(p_3)^{t_{cur}-1} = 100$. rp_0 is the nearest neighbor point dominating p_3, and rp_1 is the nearest neighbor point dominated by p_3. Thus, its upper-bound score is $100 + 8 - 6 = 102$, and its lower-bound score is $100 + 4 - 10 = 94$.

Note that newly generated data points that are forwarded to H do not have their scores. In this case, we give them to the minimum score of data points dominating them for avoiding false negatives. In addition to this, if the newly generated data points are not dominated by others, i.e., skyline, we give them to the same score of the first rank data point. Based on the above approach, we give a lemma, which is revised version of Lemma 1, and can handle the case where some top-k dominating data points expire.

Lemma 2 *Let lb_k be the kth highest lower-bound score at t_{cur}. Data points with upper-bound scores less than lb_k cannot be the top-k dominating data at t_{cur}.*

Proof This lemma is essentially the same as Lemma 1, so we omit the proof. □

To achieve this idea, H and each data site shares the positions of reference points, and each data site S_i forwards $T_i = < n_{rp_0}^{t_{cur}}, \ldots, n_{rp_{r-1}}^{t_{cur}} >$ to H every time-stamp. H merges these numbers and obtains a list of the tuple of $< n_{rp_0}, n_{rp_1}, \ldots, n_{rp_{r-1}} >$, as shown in Fig. 2b. H maintains W tuples to see the number of generated and expired data points dominated by the reference points, which enables H to reduce $|CAND|$ through tightly upper-bounding and lower-bounding scores of active data points. The scores of data points that are not broadcasted are set as their upper-bound scores. Due to the fact that the score of a data point is never less (larger) than its lower-bound (upper-bound), *CAND* always contains *TOPK*, i.e., no false negatives occurs.

4.2.3 Discussion

We here discuss the behind idea of how to place the reference points in our algorithm. First, obtaining the entire data distribution is clearly costly and impractical. In addition, the important requirements are that (i) a given data point p needs a reference point dominating p (and a reference point dominated by p), and (ii) any data distribution can leverage reference points efficiently. An intuitive way of distributing reference points is a grid-based approach, that is, the bottom left point of each grid cell is the reference point. This approach achieves the above requirements and tightens upper-bound and lower-bound scores of data points. However, it requires many reference points to be employed, which incurs large communication cost for forwarding $n_{rp_j}^{t_{cur}}$ and also increases storage cost. We therefore consider how to satisfy the above requirements with low overhead. To start with, the origin point has to be a reference point to satisfy the requirement (i), since all data points are dominated by the origin point. Next, to tighten the lower- and upper bound scores of p_i, n_{rp_j} in Eqs. (5) and (6) should be a close value to the number of data points dominated by p. Placing the reference points diagonally from the origin point, i.e., our approach, is a simple but efficient approach to achieve this with low overhead, and functions well for any data distribution. In representative data distributions such as uniform, correlated, and anti-correlated [3], for example, the nearest reference point dominating p can be $rp_j (j > 0)$ with high probability. We obtain less n_{rp_j} and tight upper-bound and lower-bound scores, since $n_{rp_0} > n_{rp_1} > \cdots > n_{rp_r}$. Furthermore, the effectiveness of this approach can be seen from the experimental results on a real dataset.

4.3 Filter points selection

We present in the previous subsection an efficient top-k dominating data computation manner with reference points that reduces unnecessary score computation. The remaining problem is to monitor k-skyband efficiently. Because the data points in k-skyband have a possibility of being the top-k dominating data at the current or future time-stamp, we have to keep it correct while avoiding unnecessary data forwarding from data sites. Therefore, we utilize filter points to avoid unnecessary data forwarding, based on the following corollary.

Corollary 1 *Given an active data point p_i, if p_i is dominated by a data point p where $\phi(p) \geq k$, p_i cannot be included in k-skyband.*

This corollary suggests that data points with d-scores $k - 1$ function well as filter points. Recall that the score of a data point implies its dominance power, i.e., data points with higher scores dominate many data points in a given dataset.

Motivated by these, given a set of active data points with d-scores $k-1$, we employ data points with the n-highest scores as filter points. Note that if there exists no data points with d-scores of $k - 1$, we can artificially generate such data points with *k-staircase* [28]. Without loss of generality, however, we assume that the above case does not occur. The (dynamic) selection of the optimal value of n is impossible or impractical because it would vary every time-stamp. Thus, we determine the value of

n when $t_{cur} = W - 1$, i.e., the first time of the top-k dominating data computation. Because we can obtain the scores of data points in the k-skyband at $t_{cur} = W - 1$, we can also compute the top-n data points with d-scores $k - 1$. Let $SKYB$ and $SKYB_j$ be the k-skyband of O and the k-skyband of O_j, respectively. To determine the value of n, we assume that the arrival times of data points in $SKYB$ and $SKYB_j$ are uniformly distributed. That is, the number of data points, generated at a given time-stamp, in $SKYB$ can be represented as $\frac{|SKYB|}{W}$. The requirement here is that the number of data forwardings at a given time-stamp should be smaller than the case of no filter points, and this condition can be represented as follows.

$$n + \left(\frac{|SKYB|}{W} + \alpha\right) < \sum_{j=1}^{m} \frac{|SKYB_j|}{W}, \tag{7}$$

where α is the number of data points not dominated by filter points but not included in $SKYB$, and this formula covers the case where all filter points expire or are replaced by others. Due to the fact of $SKYB \subseteq \bigcup_m SKYB_j$, the set of unnecessary data points is $\bigcup_m SKYB_j \setminus SKYB$, and at $t_{cur} = W - 1$, H has both $SKYB$ and $SKYB_j (1 \leq j \leq m)$ as explained in Sect. 4.2. Then, H repeatedly computes the following equation with n incrementally increasing by one.

$$dom = |\ p_{i'} \in (\cup_m SKYB_j \setminus SKYB) \mid fp_i \prec p_{i'}\ |, \tag{8}$$

where fp_i is a filter point, i.e., one of the top-n data with d-score of $k - 1$, and dom means the number of unnecessary data points dominated by filter points. We can conclude that the value of n minimizing the value of $(n - \frac{dom}{W})$ is the sufficient size for the filter point set because such a value of n minimizes the left side value of Eq. (7). H broadcasts the filter points to all data sites, and each data site identifies unnecessary data points to forward with the filter points. Note that the memory space requirement for the filter points is only $\mathcal{O}(n)$.

4.4 Query processing

We are now ready to present the filter-based algorithm, which is illustrated by Algorithm 1. Recall that the first time when $TOPK$ is computed is $t_{cur} = W - 1$, and n is also computed at this time, as described in Sect. 4.3. Below, we explain how to update $TOPK$ for a given time-stamp in the filter-based algorithm.

Each data site S_i first identifies O'_i, a set of data objects that are not dominated by n filter points (line 2), and counts the number of newly generated data dominated by the reference points (line 3). Then, S_i forwards O'_i and the counts set, T_i, to H (line 4). When H receives O'_i and T_i from S_i, H stores O'_i in O_H and merges the counts set, as described in Fig. 2 (lines 8–11). Based on the received information, H computes the lower- and upper-bound scores of data points in the k-skyband (line 16), and identifies $CAND$ (line 18), as described in Sect. 4.2. After that, this algorithm computes the top-k dominating data from $CAND$ if $|CAND| > k$ (lines 19–20). This algorithm then

Algorithm 1: Filter-based algorithm

```
1  /* procedure of a data site $S_i$ */
2  $O'_i \leftarrow \{p \mid \forall fp \in F \not\prec p\}$ // $F$ is a set of filter points, and $p$ has never been forwarded.
3  $T_i \leftarrow < n^{t_{cur}}_{rp_0}, n^{t_{cur}}_{rp_1}, \cdots, n^{t_{cur}}_{rp_{r-1}} >$
4  Forward $< O'_i, T_i >$ to $H$
5  if $S_i$ receives a set of filter points $F'$ then
6      $F \leftarrow F \cup F'$

7  /* procedure of $H$ */
8  if $H$ receives $< O'_i, T_i >$ then
9      $O_H \leftarrow O_H \cup O'_i$
10     for $j = 0$ to $r - 1$ do
11         $T_H.n^{t_{cur}}_{rp_j} \leftarrow T_H.n^{t_{cur}}_{rp_j} + T_i.n^{t_{cur}}_{rp_j}$ // $T_H.n^{t_{cur}}_{rp_j}$ is initialized at 0.

12 if $H$ receives the above tuples from all data sites then
13     $SKYB \leftarrow \{p \mid \phi(p) < k, p \in O_H\}$
14     $CAND \leftarrow SKYB$
15     if $t_{cur} \geq W$ then
16         Compute the lower- and upper-bound scores of $\forall p \in SKYB$ //Equations (5) and (6)
17         $lb_k \leftarrow$ the $k$th highest lower-bound score
18         $CAND \leftarrow \{\forall p \in SKYB \mid$ the upper-bound score of $p$ is not less than $lb_k\}$
19     if $|CAND| > k$ then
20         DIST-SCORE-COMP($CAND$) //broadcast $CAND$
21     $F' \leftarrow \{\forall p_i \mid \phi(p_i) = k - 1\}$
22     if $t_{cur} = W - 1$ then
23         Compute $n$ filter points $fp$ from $F'$
24         Broadcast the set of $fp$
25     if $t_{cur} \geq W$ then
26         Update filter points $fp$ from $F'$
27         Broadcast the set of updated $fp$
28     return $TOPK$
```

updates filter points and broadcasts them (lines 26–27). (When $t_{cur} = W - 1$, this algorithm initializes and broadcasts n filter points at lines 23–24.)

We below show an example of the filter-based algorithm by using Fig. 2.

Example 8 Assume $k = 2$ and $m = 2$, i.e., we have H, S_1, and S_2. Assume further that $t_{cur} \geq W$ and the filter-based algorithm has computed n as 2. At $t_{cur} - 1$, k-skyband is $\{p_1, p_2, p_3, p_4, p_5, p_6\}$, which are illustrated in Fig. 2b, and $n = 2$ filter points are p_2 and p_5 which have been broadcasted to S_1 and S_2. Then at t_{cur}, $\{p_{11}, p_{13}, p_{15}, p_{17}\}$ are generated at S_1, and $\{p_{12}, p_{14}, p_{16}, p_{18}\}$ are generated at S_2. Because p_{12}, p_{13}, p_{14}, p_{15}, and p_{18} are dominated by p_2 or/and p_5, they are not forwarded to H. S_1 forwards $O'_1 = \{p_{13}, p_{17}\}$ and $T_1 =< 4, 2, 1, 1 >$ to H, while S_2 forwards $O'_2 = \{p_{16}\}$ and $T_2 =< 4, 2, 1, 0 >$ to H. Note that p_{16} and p_{17} are not contained by k-skyband since p_{16} (p_{17}) is dominated by p_3 (p_6) and p_4. Also, since p_{13} is dominated by p_1, its scores is set as 52. Then H computes the lower- and upper-bound scores of $\{p_1, p_2, p_3, p_4, p_5, p_6, p_{13}\}$. We here have the second highest lower-bound score,

$lb_2 = 89$, so, at t_{cur}, $CAND = \{p_3, p_4, p_6\}$. We compute the exact scores of them by DIST- SCORE- COMP($CAND$), then we obtain $TOPK$.

5 Cache-based solution

Although the filter-based algorithm functions well for both k-skyband monitoring and top-k dominating data computation, its filtering power may be limited since the filter-algorithm utilizes only n data points with d-scores $k-1$ among $SKYB$ as filter points. That is, if H and data sites share $SKYB$, the filtering power can be enhanced. Recall that H needs to broadcast $SKYB$ when $t_{cur} = W - 1$ in Algorithm 1. Hence data sites can obtain (and cache) $SKYB$ when $t_{cur} = W - 1$, and need the incremental maintenance of $SKYB$ when $t_{cur} \geq W$ for cache consistency. If data distribution does not change quickly [15], we can assume that the change of k-skyband distribution is not so dynamic. This means that we can obtain a gain of enhanced filtering over the cache maintenance overhead. In such an environment, we employ a simple cache-based strategy to enhance k-skyband monitoring. The memory space requirement for this algorithm is $\mathcal{O}(|SKYB|)$, and note that the algorithm to compute the top-k dominating data is essentially the same as the filter-based algorithm.

5.1 Query processing

Algorithm 2 illustrates the general framework of the cache-based algorithm. Lines 2–4 and 7–11 are basically the same as Algorithm 1. Note that $\phi(p, SKYB \cup O_i)$ means that the d-score of p is calculated in the dataset of $SKYB \cup O_i$. When $t_{cur} = W - 1$, after calculating the global k-skyband, H broadcasts the k-skyband to all data sites, and the data sites cache the k-skyband (line 16). At each time-stamp $t_{cur} \geq W$, given cached k-skyband and O_j, each data site S_j calculates d-scores of its own active data points (line 2). If the data points with d-score less than k have not been forwarded yet, S_j forwards the data points to H (line 4), and H updates the global k-skyband, then broadcasts data points newly included in the k-skyband, to keep cache correct (lines 5–6 and line 19). Since the k-skyband is always shared in this algorithm, it is unnecessary to broadcast data points themselves to compute their scores. If data sites can see the identifiers of data points, data sites can compute their (local) scores. Thus, in this algorithm, when $|CAND| > k$, H broadcasts the set of the identifiers of data points included in $CAND$ to reduce communication cost (lines 22–23).

Example 9 Assume the same situation with Example 8. Since S_1 and S_2 cache k-skyband, i.e., $\{p_1, p_2, p_3, p_4, p_5, p_6\}$, the only forwarded data point is p_{13}. Then k-skyband is $\{p_1, p_2, p_3, p_4, p_5, p_6, p_{13}\}$, so H broadcasts p_{13} for cache consistency. After that, $CAND$ is computed as well as Example 8, and $TOPK$ is provided as well.

5.2 Analysis

We analyze the performance of the filter-based algorithm and the cache-based algorithm, and discuss how to select the algorithms for a given environment. The following

Algorithm 2: Cache-based algorithm

```
1  /* procedure of a data site $S_i$ */
2  $O_i' \leftarrow \{p \mid p \in O_i, \phi(p, SKYB \cup O_i) < k\}$ //p has never been forwarded.
3  $T_i \leftarrow < n_{rp_0}^{tcur}, n_{rp_1}^{tcur}, \cdots, n_{rp_{r-1}}^{tcur} >$
4  Forward $< O_i', T_i >$ to $H$
5  if $S_i$ receives a set of data points O' then
6      $SKYB \leftarrow SKYB \cup O'$ //cache maintenance

7  /* procedure of $H$ */
8  if $H$ receives $< O_i', T_i >$ then
9      $O_H \leftarrow O_H \cup O_i'$
10     for $j = 0$ to $r - 1$ do
11         $T_H.n_{rp_j}^{tcur} \leftarrow T_H.n_{rp_j}^{tcur} + T_i.n_{rp_j}^{tcur}$ //$T_H.n_{rp_j}^{tcur}$ is initialized at 0.

12 if $H$ receives the above tuple from all data sites then
13     $SKYB \leftarrow \{p \mid \phi(p) < k, p \in O_H\}$
14     $CAND \leftarrow SKYB$
15     if $t_{cur} = W - 1$ then
16         Broadcast $SKYB$
17     if $t_{cur} \geq W$ then
18         Compute the lower- and upper-bound scores of $\forall p \in SKYB$ //Equations (5) and (6)
19         Broadcast $O'$ //$O'$ is a set of data newly included in $SKYB$
20         $lb_k \leftarrow$ the $k$th highest lower-bound score
21         $CAND \leftarrow \{\forall p \in SKYB \mid$ the upper-bound score of $p$ is not less than $lb_k\}$
22     if $|CAND| > k$ then
23         DIST-SCORE-COMP($CAND$) //broadcast the set of data identifiers in
           $CAND$
24     return $TOPK$
```

theorem shows the performance differences in terms of data forwarding from data sites and top-k dominating data computation.

Theorem 1 *In terms of communication cost incurred by data forwarding from data sites, the cache-based algorithm shows equal or better performance than the filter-based algorithm. Also, in terms of communication and computation costs incurred by the top-k dominating data computation, the cache-based algorithm shows better performance than the filter-based algorithm.*

Proof Let P be a set of newly generated data points at the current time-stamp. Given FP, which is a set of filter points, and $SKYB$, we have a property that if $\exists fp \in FP$ where $fp \prec p$ for $\forall p \in P$, $\exists p' \in SKYB$ where $p' \prec p$ for $\forall p \in P$, because $FP \subseteq SKYB$. Also, even if $\exists p \in P$ where $fp \nprec p$ for $\forall fp \in FP$, it is possible that $\exists p' \in SKYB$ where $p' \prec p$. That is, the cache-based algorithm can avoid unnecessary data forwarding more than the filter-based algorithm, i.e., equal or better performance in terms of the communication cost for data forwarding from data sites. Next, let s_{id} and s_{point} be the size in bytes of data identifier and data point, respectively. The communication cost for the top-k dominating data computation is incurred by $CAND$ (or corresponding identifiers) broadcast and score forwarding from data sites. Then, such communication cost incurred by the filter-based algorithm, $cost^f$, can be simply assumed as

$$cost^f = |CAND| \cdot (s_{point} + m \cdot s_{score}),$$

where s_{score} is the size in bytes of scores of data points. Also, the communication cost for the top-k dominating data computation incurred by the cache-based algorithm, $cost^c$, can be

$$cost^c = |CAND| \cdot (s_{id} + m \cdot s_{score}).$$

CAND selection algorithm is the same between the filter- and cache-based algorithms, so $|CAND|$ is not different from each other and it is obvious $s_{id} < s_{point}$. Therefore we have that $cost^f > cost^c$. The difference of communication cost between the filter-based and cache-based algorithms becomes large, particularly in the case of large $|CAND|$. In such a case, the filter-based algorithm performs slower top-k dominating data computation due to transfer delay. The above fact shows the truth of Theorem 1. □

The other cost incurred by our algorithms is the maintenance cost of the filter and the k-skyband caching, but they cannot be comparable because the need for such maintenance depends on data generations and expirations. If k-skyband changes dramatically at every time-stamp, the cache-based algorithm may lose the advantages described above. If k-skyband does not change frequently, on the other hand, we have low maintenance cost for the cache consistency, meaning that the cache-based algorithm clearly outperforms the filter-based algorithm. However, the memory space requirement of the cache-based algorithm, $\mathcal{O}(|SKYB|)$, is larger than that of the filter-based algorithm, $\mathcal{O}(n)$. It can therefore be a trade-off relationship between communication efficiency and storage complexity, and if applications prefer small memory consumption, the filter-based algorithm might be better. This is because our experimental results show that the cache-based algorithm shows better performance than the filter-based algorithm, but the filter-based algorithm is very competitive with the cache-based algorithm in terms of both communication cost and computation time.

6 Sampling-based approximation

The algorithms introduced in Sects. 4 and 5 continuously monitor *exact* top-k dominating data efficiently. However, some applications may not need perfect accuracy of the set of top-k dominating data [2,15,26], and *approximate* result satisfies such applications. We can reduce both communication and computation costs by sacrificing accuracy of *TOPK*, if approximate result is sufficient.

Interestingly, our top-k dominating computation algorithm can be easily extended to the approximate computation algorithm with probabilistic guarantee. Given a user-specified error rate ϵ, we can guarantee at least $1 - \epsilon$ accuracy of *TOPK*, if satisfying the following formula.

Algorithm 3: Sampling algorithm

```
  Input: $\epsilon$ and the set of samples of the variation of the $k$th highest score
1 /* procedure of $H$ */
2 if $\frac{k}{|CAND|} < 1 - \epsilon$ then
3     $\tau \leftarrow$ ScoreEstimation() //the $k$th highest score estimation
4     $CAND \leftarrow$ CandidateSelection($SKYB$, $\tau$)
5     if $\frac{k}{|CAND|} < 1 - \epsilon$ then
6         DIST- SCORE- COMP($CAND$)
```

$$\frac{k}{|CAND|} \geq 1 - \epsilon \tag{9}$$

In the above formula, the left-hand side means the precision of *TOPK*. Hence, if the above formula is satisfied, H does not need to broadcast *CAND* (or the set of its corresponding data identifiers) and simply reports k data points with the highest lower-bound (or upper-bound) scores, which reduces communication cost as well as computation cost. However, such a case are limited, particularly if k is small or ϵ is small. In addition to this, ϵ is practically specified as an indication, i.e., users do not care as long as the accuracy is *around* $1 - \epsilon$. These motivate us to develop more efficient heuristic algorithm to compute approximate top-k dominating data, and we propose a sampling-based top-k dominating data computation algorithm. Note that this algorithm can be employed by both Algorithms 1 and 2.

Our idea is based on statistics and to estimate the kth highest score at t_{cur} from samples and reduce $|CAND|$. Every sequential time-stamps when H computes the exact scores of data points in *CAND*, H stores the *variation* of the kth highest score as a sample. Exploiting the samples, we estimate the variation from the kth highest score in the previous time-stamp. Given the set of samples V, we have a sample mean $\bar{v} = \frac{\sum_{i=1}^{V} v_i}{V}$ and an unbiased variance $U^2 = \frac{1}{V-1}\sum_{i=1}^{V}(v_i - \bar{v})^2$, where v_i is a variation value. From interval estimation, we obtain the estimated population mean μ_{est} as follows.

$$\bar{v} - \beta \cdot \frac{U}{\sqrt{V}} \leq \mu_{est} \leq \bar{v} + \beta \cdot \frac{U}{\sqrt{V}}$$

The value of β is dependent on the confidence δ and V (if $V < 30$, t-distribution should be used, otherwise normal distribution is available). We use $\bar{v} - \beta \cdot \frac{U}{\sqrt{V}}$ as μ_{est}, i.e., the variation from the previous kth highest score, because we have $\mu_{exact} \geq \mu_{est}$ with the probability of δ from probability theory. Exploiting μ_{est}, H calculates the estimated kth highest score, and data points with upper-bound scores less than the estimated score are eliminated from *CAND*. Rank orders of data points are determined by their lower-bound scores in this algorithm since the upper-bound scores of data points not included in the previous *TOPK* tend to be large, which might lead to violation of the exact rank order.

Algorithm 3 illustrates the overview of the procedures of the sampling-based top-k dominating data computation. This procedures replace the procedures of lines 19–

20 of Algorithm 1 and lines 22–23 of Algorithm 2. If the precision is less than the user-specified accuracy, H performs the operation of the kth highest score estimation described above (line 3), then re-calculates *CAND* (line 4). In the case where the precision is still less than the user-specified accuracy, H broadcasts *CAND* (or the set of its corresponding data identifiers) to all data sites for the exact score computation.

This approximate algorithm brings less communication cost to monitor the top-k dominating data and avoids the top-k dominating data computation. Due to the approximate *CAND* selection, we cannot guarantee the perfect accuracy of *TOPK*. We thus prevent a decrease of accuracy by the following algorithm design. Data points with higher upper-bound scores are probable to be included in *CAND*, and they are the most promising data points for the top-k dominating data. Note that the kth highest score at the previous time-stamp may be an estimated value, which may decrease the accuracy. However, if the approximation is executed in sequential time-stamps, the estimated kth highest score at t_{cur} is likely to be less than upper-bound scores of data points. That is, $|CAND|$ is likely to be large, and the condition of "$\frac{k}{|CAND|} \geq 1 - \epsilon$" becomes difficult to be satisfied. In this case, the exact score computation is executed and we would obtain a sample of the variation.

7 Experimental evaluation

In this section, we provide an experimental evaluation on the performance of our algorithms. The evaluated algorithms are as follows.

- CEN: the naive algorithm introduced in Sect. 3.3.
- SKYB: each site forwards data objects that newly become ones of the local k-skyband to H, and H broadcasts the global k-skyband, then computes the top-k dominating data in the same way as the proposed algorithm at every time-stamp.
- TD-F: the algorithm proposed in Sect. 4.
- TD-C: the algorithm proposed in Sect. 5.
- TD-S: the algorithm with the cache-based k-skyband monitoring and the approximate top-k dominating data computation algorithm proposed in Sect. 6.

As introduced in Sect. 1, this is the first work for continuous top-k dominating query processing in distributed data streams, so there is no existing solution for this problem. Hence we compare our algorithms with the centralized solution (CEN) and the simple solution (SKYB). All algorithms were implemented in C++, and all experiments were conducted on a PC with 3.4GHz Intel Core i7 processor and 32GB RAM.

7.1 Datasets

We conducted the experiments on both synthetic and real datasets. We generated synthetic d-dimensional datasets of uniform (UN), clustered (CL), and anti-correlated (AC) distributions for careful evaluation, which are often used to evaluate performances of queries related to dominance relationships. UN includes random points in a space $[0, 1000]^d$. In CL, we draw a coordinate of its centroid of a cluster on the d-dimensional space $[0, 1000]^d$. The value on each dimension independently follows

Gaussian distribution with a variance of 50. Finally, we generated AC with reference to [3] and AC includes points whose coordinates are anti-correlated. That is, points which are good in one dimension are bad in one or all other dimensions.

For real data stream evaluation, as existing works employed [21,41], we used a network traffic dataset. This dataset is Cdad [7], which was collected in a distributed environment (IBM Watson). Cdad has 2,726,561 snmp network usage data, and in terms of data attributes, we used the first four attributes, `snmpInPkts`, `snmpOutPkts`, `ipIn`, and `ipOut`.

7.2 Distributed setting

The central server H can communicate with m distributed data sites, and we assume 100[Mbps] as network transfer rate on connections between H and a data site. We also varied the network transfer rate because this parameter relates to score computation time, but the results are quantitatively-similar, hence we do not report in this paper.

7.3 Evaluation criteria

We measured following criteria.

- Total communication cost: the communication cost incurred by the top-k dominating data monitoring. Data attribute values on each dimension are 8 bytes, and other information, such as identifiers, arrival time, etc., are all 4 bytes.
- Top-k data calculation and data transmission costs: the communication costs incurred by the top-k dominating data evaluation and the k-skyband maintenance, respectively. Note that CEN computes the top-k dominating data objects at H locally, so for CEN, this metric is not available. In addition to this, the data transmission costs of TD-S is the same as that of TD-C.
- Computation time: the average time for the top-k dominating data computation.
- Accuracy: the average recall of *TOPK*, i.e., the number of actual top-k dominating data returned by TD-S divided by k.

7.4 Experiments using synthetic data

In the experiments using the synthetic datasets, each site generates d-dimensional data objects every time-stamp, and the number of data objects generated by a site follows Gaussian distribution with a mean of g/m and a variance of 10, where g is the data generation rate at a time-stamp. Table 2 summarizes the parameters used in the experiments and the bold values are default values. Note that the window length, W, is set so that the window contains around 1 million data by default. From preliminary experiments, we set 25 and 10 as r and l, respectively.

The simulation length, that is, the number of time-stamps is 400, and the first W time-stamps are not scope of the evaluation. In this experiment, TD-S does not perform the approximate top-k dominating data computation in the first 20 time-stamps for sample collection. We varied the number of time-stamps, and have confirmed that

Table 2 Configuration of parameters

Parameter	Values
Value of k	5, **10**, 15, 20, 25, 30
Data generation rate, g [$\times 10^3$]	5, **10**, 15, 20, 25, 30
Window length, W	10, 50, **100**, 150, 200
Dimensionality, d	2, **3**, 4, 5, 6
Data distribution	**UN**, CL, AC
#sites, m	10, 25, **50**, 100
Error rate, ϵ	**0.1**, 0.2, 0.25, 0.3
Confidence, δ	**0.9**

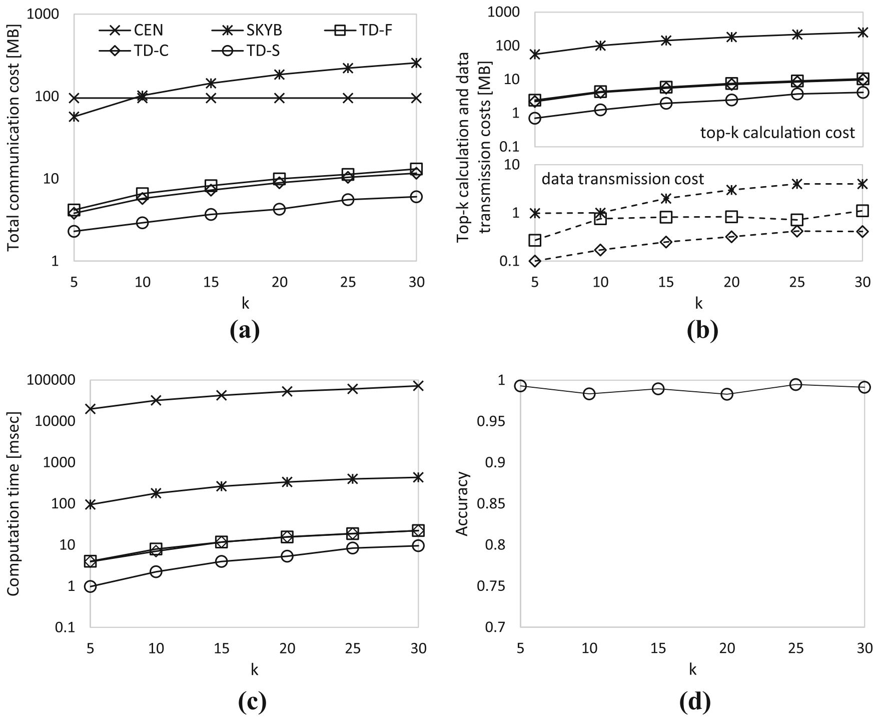


Fig. 3 Impact of k. **a** Total communication cost. **b** Top-k calculation and data transmission costs. **c** Computation time. **d** Accuracy

the number of time-stamps 400 is sufficient for averaging, i.e., computation time and accuracy do not vary quantitatively when the number of time-stamps increases. In addition to this, the difference in performance between our algorithms and the comparison algorithms (CEN and SKYB) also do not vary, rather, in terms of communication cost, the difference becomes larger, as the number of time-stamps increases.

7.4.1 Varying k

We examined the impact of k, and Fig. 3 shows the result with varying k. Although the total communication cost and the computation time increase as k increases in our

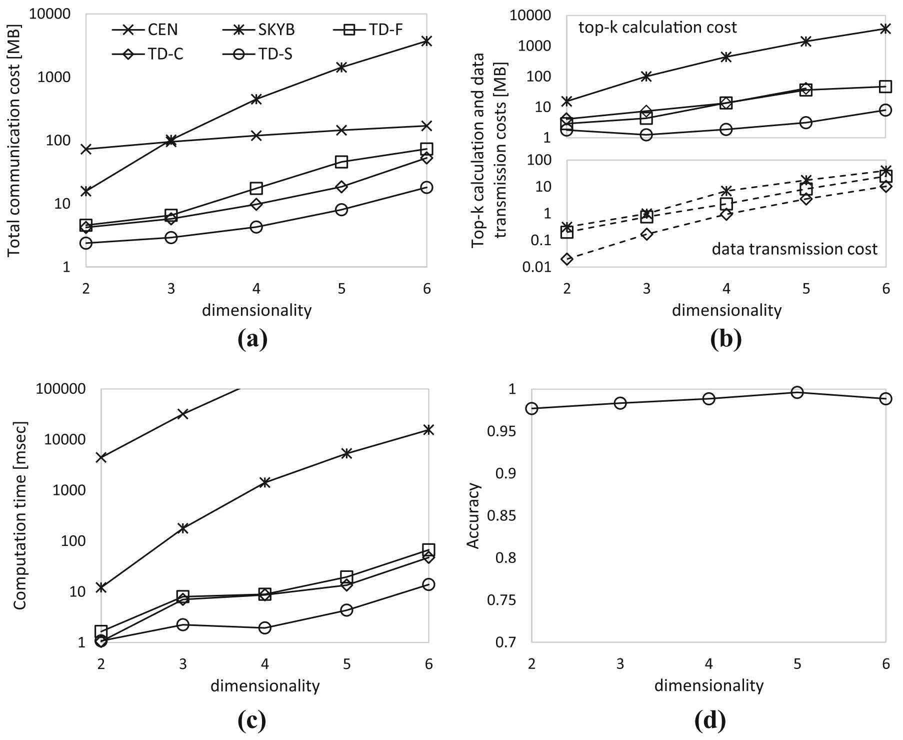


Fig. 4 Impact of dimensionality. **a** Total communication cost. **b** Top-k calculation and data transmission costs. **c** Computation time. **d** Accuracy

algorithms, they significantly outperform CEN and SKYB w.r.t. all criteria, e.g., around one order of magnitude better in terms of total communication cost (Fig. 3a). For CEN, k does not affect the communication cost because all data points are forwarded to H, and Fig. 3b shows that broadcasting k-skyband at every time-stamp is inefficient. In Fig. 3b, TD-F reduces the communication cost for data forwarding compared with SKYB, which stems from Eq. (7). TD-C further reduces the communication cost compared with TD-F because data points in cache dominate more unnecessary data points. We can see the impact of parallel top-k dominating data computation from Fig. 3c, and SKYB reduces computation time one order of magnitude than that of CEN. Our algorithms enhance its impact more due to their efficient *CAND* selection, and their computation times are quite short. TD-S shows the lowest total communication cost and computation time while keeping highly accurate query result as seen in Fig. 3d.

7.4.2 Varying d

We next examined the impact of d, and Fig. 4 shows the result with varying d. As shown in Fig. 4a, our algorithms keep lower total communication cost even when d is large, e.g., TD-C shows 50 % lower than CEN. Because it is not easy to dominate other data points if d is large, the communication cost for data forwarding becomes large as d

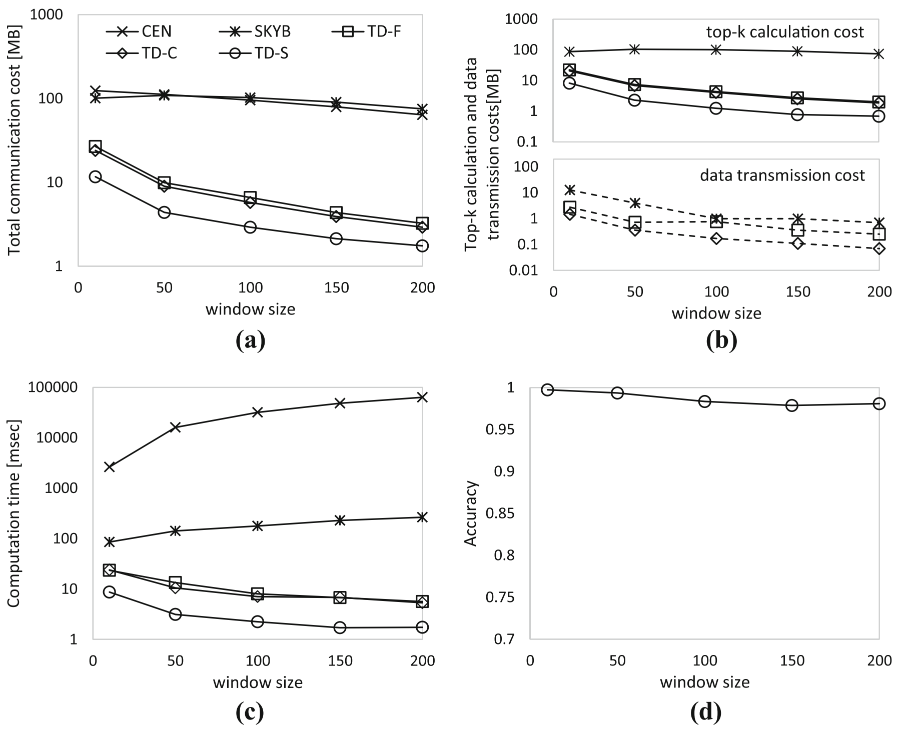


Fig. 5 Impact of window size. **a** Total communication cost. **b** Top-k calculation and data transmission costs. **c** Computation time. **d** Accuracy

increases. The effect of reference points, however, still keeps, avoiding redundant data broadcast and computation cost as Fig. 4c shows. Moreover, from Fig. 4c, we can see that the computation times of our algorithms are significantly faster than that of CEN, which is practically meaningful for many application requiring real-time monitoring. In the cases of large d, the number of *CAND* broadcast events is larger compared with small d, so the accuracy of TD-S is high (see Fig. 4d).

7.4.3 Varying W

We examined the impact of W because the number of active data points depends on the window length, and Fig. 5 shows the result with varying W. Note that the reason why the total communication cost decreases as W becomes large is the limitation of simulation length (Fig. 5a). Although the number of active data points increases as W becomes large, our algorithms still keep their efficiency, and even improves efficiency. Because small W leads to quick data expiration, the maintenance costs for filter, cache, and *TOPK* become larger as shown in Fig. 5b, c. That is, in the case of small W, $|CAND|$ is likely to be large, which makes difficult for TD-S to satisfy Eq. (9). Therefore, TD-S performs the exact score computation, resulting in quite high accuracy of TD-S.

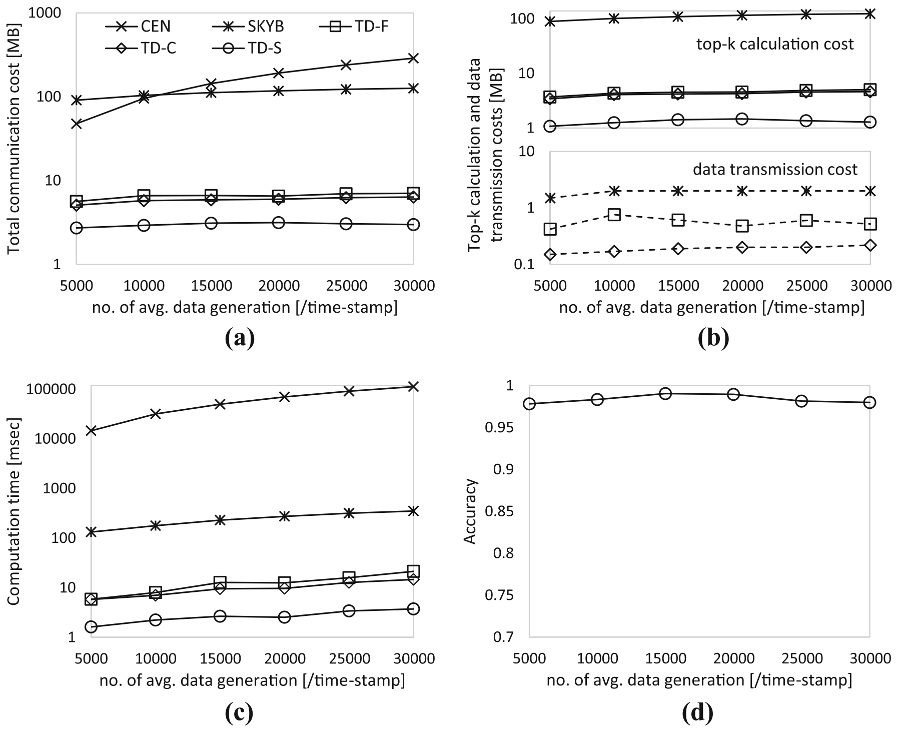


Fig. 6 Impact of data generation rate. **a** Total communication cost. **b** Top-k calculation and data transmission costs. **c** Computation time. **d** Accuracy

7.4.4 Varying g

To investigate whether our algorithms function well even in an environment where data generation rate is high, we examined the impact of g, and Fig. 6 shows the result with varying g. Fig. 6a shows that for CEN and SKYB, high-rate data generation incurs high communication cost, while our algorithms are not basically affected. This trend can be seen also in Fig. 6b, but computation time becomes longer in the cases of high data generation rate (Fig. 6c) because the cardinality of local databases becomes large, which needs larger number of dominance relationship checks. From Fig. 6d, we can see that g does not affect the accuracy of TD-S.

7.4.5 Varying m

The number of data sites also affect the performances of our algorithms because H needs to broadcast *CAND* to all data sites and the number of newly generated data points dominated by reference points are also forwarded from all data sites. Thus we examined the impact of m, and Fig. 7 shows the result with varying m. Figure 7a shows that as m increases, the total communication costs of our algorithms increase, while

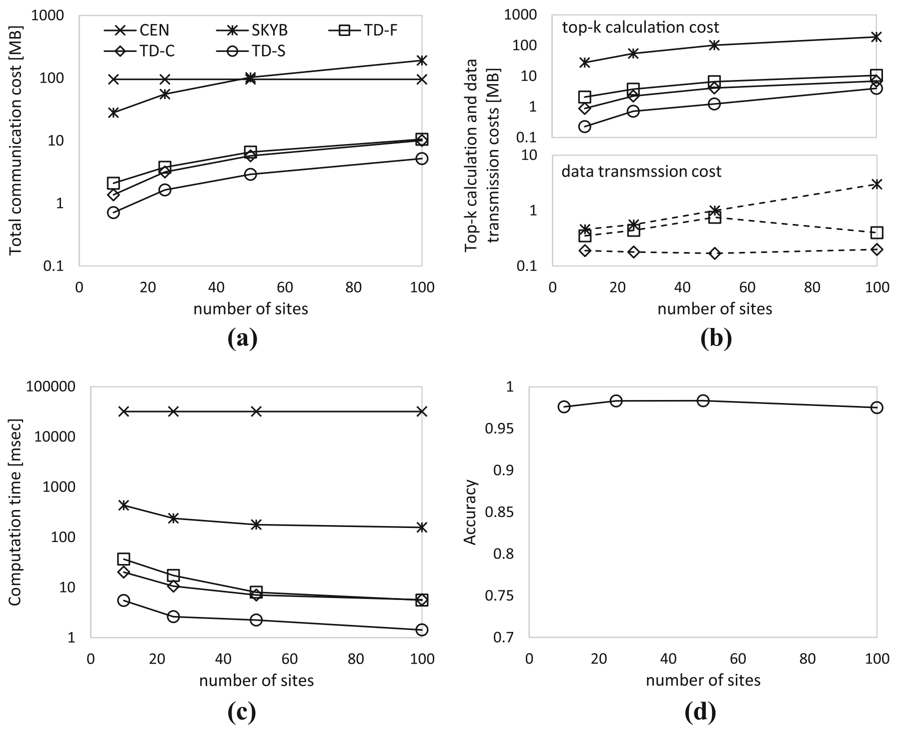


Fig. 7 Impact of #sites. **a** Total communication cost. **b** Top-k calculation and data transmission costs. **c** Computation time. **d** Accuracy

much smaller than those of CEN and SKYB. This is because the communication costs for forwarding the number of newly generated data points dominated by reference points and for top-k data computation increase as *m* increases (shown in Fig. 7b). Meanwhile, computation time becomes faster as *m* increases. Since our algorithms compute scores of data points in parallel, the score computation time per a data site becomes faster as the entire dataset is more distributed. The accuracy of *TOPK* in TD-S is not affected by *m*, and keeps high.

7.4.6 Varying data distribution

We also evaluated our algorithms with varying datasets, and Fig. 8 shows the result. From Fig. 8a, b, we can see that our algorithms outperform CEN and SKYB in terms of communication cost even in the case of AC in which data points are likely not to be dominated. Also, the computation times of our algorithms are quite shorter than those of CEN and SKYB. Multi-dimensional grid functions well particularly in the case of CL, and the computation time of TD-S shows less than half of that of TD-C, while keeping high accuracy as shown in Fig. 8d.

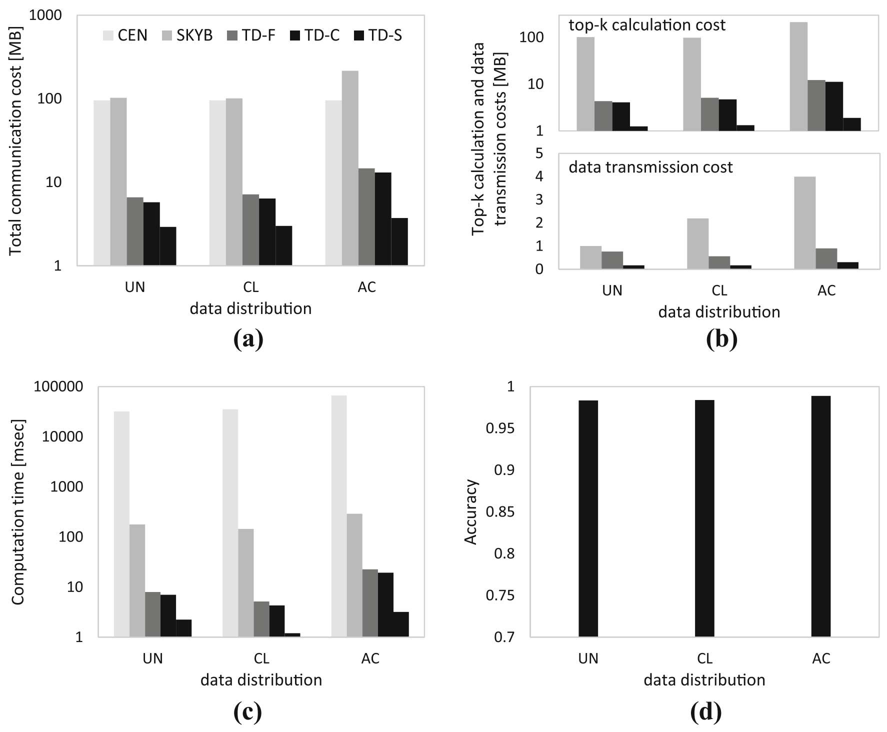


Fig. 8 Impact of data distribution. **a** Total communication cost. **b** Top-k calculation and data transmission costs. **c** Computation time. **d** Accuracy

7.4.7 Varying ϵ

In TD-S, ϵ directly affects the accuracy of *TOPK*, thus we examined the impact of ϵ, and Fig. 9 shows the result with varying ϵ. As shown in Fig. 9a, b, the communication cost of TD-S decreases as ϵ increases, which means that the number of avoiding *CAND* broadcasting increases as ϵ increases. Therefore, the computation time also becomes faster with the increase of ϵ. Figure 9d shows the trade-off relationship between accuracy and communication cost (and computation cost). However, the accuracy is always higher than $1 - \epsilon$.

7.5 Experiment using real data

Finally, we evaluated the performances of our proposed algorithms by the real dataset. We set 25 as m, and each stream data is assigned to one of the 25 data sites by hashing [7]. W, ϵ, δ, l, and r are 5000, 0.1, 0.9, 10, and 20, respectively.

This experiment was conducted by varying k, and Fig. 10 shows the result. As the results of the experimental using the synthetic datasets show, the performance of CEN is poor. So, in Fig. 10, we omit the result of CEN. Note that we confirmed that TD-F, TD-C, and TD-S outperform CEN.

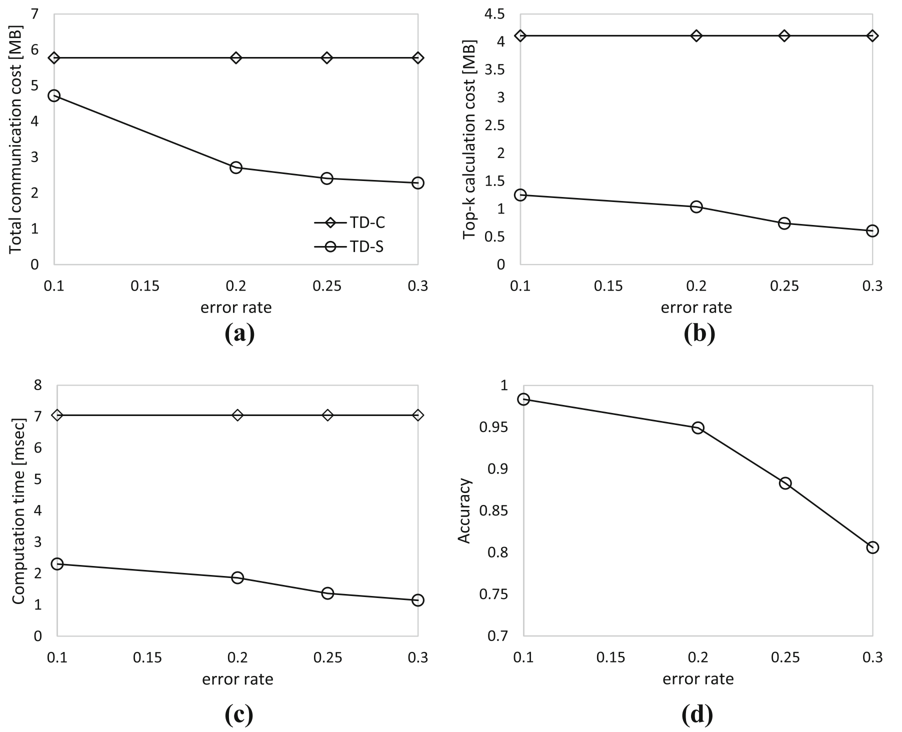


Fig. 9 Impact of error rate. **a** Total communication cost. **b** Top-k calculation and data transmission costs. **c** Computation time. **d** Accuracy

We observed that the size of *k*-skyband is small, thus the total communication costs of our algorithms are similar with that of SKYB when *k* is small, as shown in Fig. 10a. However, we can see that TD-F and TD-C scale better than SKYB, and TD-S furthermore scales better that TD-F and TD-C. We can see, from Fig. 10b, that the reference points function well and eliminate unqualified data for *TOPK*. In view of the data transmission cost, the filter and cache approaches efficiently reduce unnecessary data forwarding compared with SKYB. Also, in terms of computation time, as with communication cost, our algorithms outperform SKYB, which can be seen in Fig. 10c. TD-S shows the shortest computation time while keeping high accuracy, as shown in Fig. 10d.

8 Conclusion

In this study, we investigated the important and challenging problem of continuous top-k dominating query processing over distributed data stream. We are the first to address this problem. Because repeating snapshot query processing incurs high communication cost as well as computation cost, which is not suitable for stream environments. We proposed efficient approaches that always maintain non-zero probability of being the top-k dominating data, and carefully select candidate data points from them through

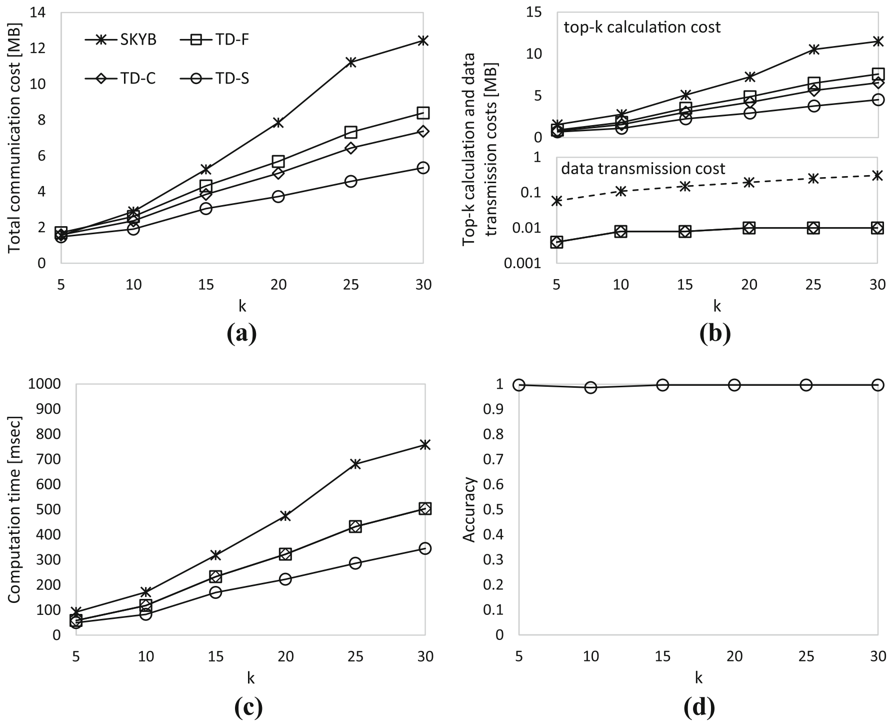


Fig. 10 Impact of *k* (real dataset). **a** Total communication cost. **b** Top-k calculation and data transmission costs. **c** Computation time. **d** Accuracy

reference points. Although approximate top-k dominating data computation sacrifices accuracy of the query result, we can reduce both communication and computation costs. We evaluated our algorithms by experiments on both synthetic and real data, and the results show that our algorithms significantly reduce both communication and computation costs. Furthermore, the experimental results show a trade-off relationship between accuracy of the query result and communication and computation costs.

One of the probable future works is to handle continuous subspace top-k dominating queries. If users are interested in only a part of attributes and there are multiple preferences, multiple subspace top-k dominating queries can be posed. Efficiently monitoring such data is an interesting problem to solve.

Acknowledgments This research is partially supported by the Grant-in-Aid for Scientific Research (A)(26240013) of MEXT, Japan, and JST, Strategic International Collaborative Research Program, SICORP.

References

1. Babcock, B., Babu, S., Datar, M., Motwani, R., Widom, J.: Models and issues in data stream systems. In: Proceedings of PODS, pp. 1–16. Springer, New York (2002)
2. Babcock, B., Olston, C.: Distributed top-k monitoring. In: Proceedings of SIGMOD, pp. 28–39. Springer, San Diego (2003)

3. Börzsönyi, S., Kossmann, D., Stocker, K.: The skyline operator. In: Proceedings of ICDE, pp. 421–430. Springer, New York (2001)
4. Chan, C.Y., Jagadish, H., Tan, K.L., Tung, A.K., Zhang, Z.: Finding k-dominant skylines in high dimensional space. In: Proceedings of SIGMOD, pp. 503–514. Springer, New York (2006)
5. Cheema, M.A., Lin, X., Zhang, W., Zhang, Y.: A safe zone based approach for monitoring moving skyline queries. In: Proceedings of EDBT, pp. 275–286. IEEE Press, Geneva (2013)
6. Das, G., Gunopulos, D., Koudas, N., Sarkas, N.: Ad-hoc top-k query answering for data streams. In: Proceedings of VLDB, pp. 183–194. ACM, New York (2007)
7. Garofalakis, M., Keren, D., Samoladas, V.: Sketch-based geometric monitoring of distributed stream queries. PVLDB **6**(10), 937–948 (2013)
8. Giatrakos, N., Deligiannakis, A., Garofalakis, M., Sharfman, I., Schuster, A.: Prediction-based geometric monitoring over distributed data streams. In: Proceedings of SIGMOD, pp. 265–276. ACM Press, Scottsdale (2012)
9. Han, X., Li, J., Gao, H.: Tdep: Efficiently Processing Top-k Dominating Query on Massive Data, pp. 1–30. Knowledge and Information Systems, Springer, Berlin (2014)
10. He, Z., Lo, E.: Answering why-not questions on top-k queries. In: Proceedings of ICDE, pp. 750–761. Springer, New York (2012)
11. Hose, K., Vlachou, A.: A survey of skyline processing in highly distributed environments. VLDB J. **21**(3), 359–384 (2012)
12. Huang, Q., Lee, P.P.: Ld-sketch: A distributed sketching design for accurate and scalable anomaly detection in network data streams. In: Proceedings of INFOCOM, pp. 1420–1428. IEEE Press, Geneva (2014)
13. Huang, Z., Lu, H., Ooi, B.C., Tung, A.: Continuous skyline queries for moving objects. IEEE TKDE **18**(12), 1645–1658 (2006)
14. Kontaki, M., Papadopoulos, A.N., Manolopoulos, Y.: Continuous top-k dominating queries in subspaces. In: Panhellenic Conference on Informatics, pp. 31–35. IEEE Press, New York (2008)
15. Kontaki, M., Papadopoulos, A.N., Manolopoulos, Y.: Continuous top-k dominating queries. IEEE TKDE **24**(5), 840–853 (2012)
16. Lee, Y.W., Lee, K.Y., Kim, M.H.: Efficient processing of multiple continuous skyline queries over a data stream. Inf. Sci. **221**, 316–337 (2013)
17. Lian, X., Chen, L.: Top-k dominating queries in uncertain databases. In: Proceedings of EDBT, pp. 660–671. IEEE Press, New York (2009)
18. Lin, X., Yuan, Y., Wang, W., Lu, H.: Stabbing the sky: efficient skyline computation over sliding windows. In: Proceedings of ICDE, pp. 502–513. IEEE Press, San Diego (2005)
19. Lin, X., Yuan, Y., Zhang, Q., Zhang, Y.: Selecting stars: the k most representative skyline operator. In: Proceedings of SIGMOD, pp. 86–95. Springer, Heidelberg (2007)
20. Lu, H., Zhou, Y., Haustad, J.: Continuous skyline monitoring over distributed data streams. In: SSDBM, pp. 565–583. Springer, Heidelberg (2010)
21. Lu, H., Zhou, Y., Haustad, J.: Efficient and scalable continuous skyline monitoring in two-tier streaming settings. Inf. Syst. **38**(1), 68–81 (2013)
22. Morse, M., Patel, J.M., Grosky, W.I.: Efficient continuous skyline computation. Inf. Sci. **177**(17), 3411–3437 (2007)
23. Mouratidis, K., Bakiras, S., Papadias, D.: Continuous monitoring of top-k queries over sliding windows. In: SIGMOD, pp. 635–646. ACM New York (2006)
24. Papadias, D., Tao, Y., Fu, G., Seeger, B.: Progressive skyline computation in database systems. ACM Trans. Database Syst. **30**(1), 41–82 (2005)
25. Papapetrou, O., Garofalakis, M.: Continuous fragmented skylines over distributed streams. In: Proceedings of ICDE, pp. 124–135. ACM Press, New York (2014)
26. Papapetrou, O., Garofalakis, M., Deligiannakis, A.: Sketch-based querying of distributed sliding-window data streams. PVLDB **5**(10), 992–1003 (2012)
27. Santoso, B., Chiu, G.: Close dominance graph: an efficient framework for answering continuous top-k dominating queries. IEEE TKDE **26**(8), 1853–1865 (2014)
28. Shen, Z., Cheema, M.A., Lin, X., Zhang, W., Wang, H.: Efficiently monitoring top-k pairs over sliding windows. In: Proceedings of ICDE, pp. 798–809. Springer, New York (2012)
29. Skoutas, D., Sacharidis, D., Simitsis, A., Kantere, V., Sellis, T.: Top-k dominant web services under multi-criteria matching. In: Proceedings of EDBT, pp. 898–909. Springer, New York (2009)

30. Sun, S., Huang, Z., Zhong, H., Dai, D., Liu, H., Li, J.: Efficient monitoring of skyline queries over distributed data streams. Knowl. Inf. Syst. **25**(3), 575–606 (2010)
31. Tao, Y., Papadias, D.: Maintaining sliding window skylines on data streams. IEEE TKDE **18**(3), 377–391 (2006)
32. Tiakas, E., Valkans, G., Papadopoulos, A.N., Manolopoulos, Y., Gunopoulos, D.: Metric-based top-k dominating queries. In: Proceedings of EDBT, pp. 415–426. IEEE Press, San Diego (2014)
33. Vlachou, A., Doulkeridis, C., Nørvåg, K., Vazirgiannis, M.: On efficient top-k query processing in highly distributed environments. In: Proceedings of SIGMOD, pp. 753–764. Springer, New York (2008)
34. Xie, X., Lu, H., Chen, J., Shang, S.: Top-k neighborhood dominating query. In: Proceedings of DASFAA, pp. 131–145. Springer, Berlin (2013)
35. Yang, D., Shastri, A., Rundensteiner, E.A., Ward, M.O.: An optimal strategy for monitoring top-k queries in streaming windows. In: Proceedings of EDBT, pp. 57–68. ACM, New York (2011)
36. Yiu, M.L., Mamoulis, N.: Efficient processing of top-k dominating queries on multi-dimensional data. In: Proceedings of VLDB, pp. 483–494. Springer, New York (2007)
37. Yiu, M.L., Mamoulis, N.: Multi-dimensional top-k dominating queries. VLDB J. **18**(3), 695–718 (2009)
38. Yu, A., Agarwal, P., Yang, J.: Processing a large number of continuous preference top-k queries. In: Proceedings of SIGMOD, pp. 397–408. ACM Press, New York (2012)
39. Zhang, W., Lin, X., Zhang, Y., Pei, J., Wang, W.: Threshold-based probabilistic top-k dominating queries. VLDB J. **19**(2), 283–305 (2010)
40. Zhang, W., Lin, X., Zhang, Y., Pei, J., Wang, W.: Progressive processing of subspace dominating queries. VLDB J. **20**(6), 921–948 (2011)
41. Zhang, Z., Cheng, R., Papadias, D., Tung, A.K.: Minimizing the communication cost for continuous skyline maintenance. In: Proceedings of SIGMOD, pp. 495–508. ACM Press, New York (2009)