DEIM Forum 2016 G8-6

Reducing Expenses of Sensor-Cloud Services for Dynamic Skyline Monitoring

Kamalas UDOMLAMLERT†, Takahiro HARA†, and Shojiro NISHIO†

† Graduate School of Information Science and Technology, Osaka University
1-5 Yamadaoka, Suita, Osaka 565-0871, Japan
E-mail: †{kamalas.u,hara,nishio}@ist.osaka-u.ac.jp

Abstract In the concept of Sensor-Cloud, many sensor networks are unified and provide Sensing as a Service to users. The users can utilize provisioning virtual sensors to perform desired tasks. The expense (cost) is based on the amount and types of data requests. In this paper, unlike other existing works, we propose a cost-minimizing framework for dynamic skyline monitoring. Our approach avoids paying the cost of all attributes by iteratively requesting the most cost-effective attributes that can identify whether a tuple is in dynamic skyline. We conduct some experiments on various real datasets. The results confirm the effectiveness compared with the baseline and the random strategy.

Key words dynamic skyline, monitoring query, sensor cloud, sensing-as-a-service

1. Introduction

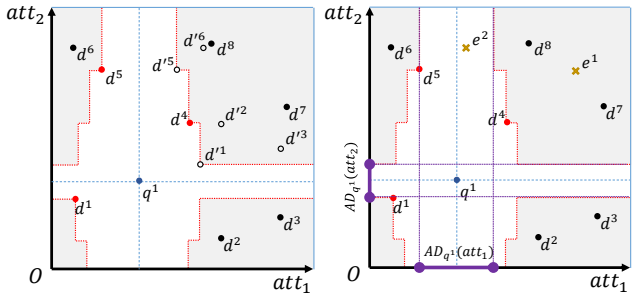
Wireless Sensor Networks (WSNs) play an important role in various fields including environmental monitoring, structure health monitoring, disaster prevention, industrial monitoring and so forth. These have been extensively researched in the research community. Because of the development of mobile phones and social networks, the concept of mobile sensing – using sensors on mobile phones as a sensing platform has been introduced. Therefore, a massive amount of information has been generated everyday, which enables many captivating applications. However, most of real sensor networks are deployed and utilized for only a specific purpose on limited locations by private organizations. According to individually-invested expensive installation cost, those resources are rarely federated and not often open to public. Moreover, in mobile sensing, mobile phone users are not willing to do sensing tasks due to privacy concerns and mobile battery constraints.

Dealing with Big Sensor Data, the idea of using cloud computing (Sensor-Cloud) aims to unify available sensors together, i.e., various existing WSNs and mobile users, and many research challenges to create such a platform have been addressed [1].

Sensing as a service (S^2aaS) [8] in Sensor-Cloud abstracts all the physical layers of sensor network integration. It provides the cloud users to effectively utilize the provisioning virtual sensors, so cloud users can perform their desire sensing tasks by using various kinds of sensors in an available wide-spread area without technical difficulties. In commercial services [4], cloud users pay the usage cost based on the amount of utilization (pay as you go) without the necessity of deploying their own infrastructure, while WSNs providers (owner) and mobile phone users may receive the rewards in the forms of rental fee or the cost of requested data.

Meanwhile, primitive tasks of WSNs are monitoring. Due to various types of sensors integrated in Sensor-Cloud, many monitoring systems take more than one attribute into account from various kinds of provided sensors to make a decision (Multiple-criteria decision-making). Hence, many complex monitoring queries for multidimensional data have been developed including top-k query, k-nearest neighbor query, skyline query and reverse skyline query. Based on the investment and production cost, different types of sensors provide different kind of data at different prices in Sensor-Cloud. Even though, the traditional monitoring methods can be used on Sensor-Cloud, none of previous works has addressed the important objective for cloud users when using Sensor-Cloud, i.e., cost minimization.

In this work, we firstly propose a cost-minimization monitoring framework for dynamic skyline queries, a useful query method for WSNs, where a traditional skyline is a special case. Given a reference point (query point) in normal situation and a data set of critical points, the system monitors periodically-sensed data from sensors (event tuple). To illustrate, an administrator monitors a combination of risk factors of flash flood, e.g., ground water level, precipitation, high tide and melting snow on a location every 30 minutes.



(a) Dynamic skyline (b) Initial anti-dominance ranges

Figure 1: The example of dynamic skyline of q^1

The standard levels of these factors are used as a reference point. If their values are close to the reference point, it shows a benign event takes place. Otherwise, if the values deviate from the reference point until they are worse than (dominated by) the given critical points, i.e., become a dynamic skyline, the administrator must look at such an event with attention. However, knowing actual values of some attributes may lead to a conclusion. For example, the readings now shows that ground water level and precipitation are very low and close to the standard levels, so, regardless of other factors, there is no chance of flood. In this case, we can save utilization cost by not requesting sensor data for such negligible factors.

Our proposed framework evaluates cost-effective measurement for each attribute to decide a plan, which sensor should be requested in order to lead to the query answer with the lowest cost. We conduct some experiments using real datasets to ensure the capability and scalability of our proposed framework.

2. Related Work

Sensor-Cloud is a new concept which is inherited from Cloud computing. Cloud computing basically provides users many XaaS (X as a service) such as Platform as a Service (PaaS). Sensor-Cloud aims to integrate and abstract many physical sensors and/or any communicating things as an infrastructure that allows users to easily utilize it with lower cost than owning WSNs. This is called Sensing as a Service (S^2aaS) [8], and some services can become commercial [4]. The comprehensive survey of architecture, advantages and provisioning applications are covered up in [1].

To draw a conclusive decision from multi-dimensional data, e.g, sensor data, many complex query processing techniques have been developed. Among these, skyline processing plays an important role in various applications such as monitoring and service discovery [9]. Most works address efficient skyline processing [7] and distributed skyline processing [3]. In this work, we firstly investigate a cost minimization problem for dynamic skyline [6], which is a superclass of traditional skyline, in Sensor-Cloud.

3. Cost Minimizing Framework

3.1 Problem Statement

Given a set of m attributes including $AT = \{att_1, att_2, \ldots, att_m\}$, and let $D = \{d^1, d^2, \ldots, d^{N_D}\}$ be a dataset of critical points in m-dimensional positive Euclidean space of attributes $\mathcal{A} = A_1 \times A^2 \times \ldots \times A^m$. $d^i \in D$ can be represented as a point $(d^i_1, d^i_2, ..., d^i_m) \in \mathcal{A}$ where $d^i_j \in A_j$, $i \in \{1, 2, ..., N_D\}$ and $j \in \{1, 2, \ldots, m\}$.

Definition 1. *(Dynamic Skyline Query) [2], [6]: Given an m-dimensional query point $q \in A$ and a dataset D, a Dynamic Skyline Query (DSQ) w.r.t. q retrieves all data points in D that are not dynamically dominated denoted by $DSQ(D, q)$. A point $d \in D$ dynamically dominates $d' \in D$ w.r.t. the query point q if (1) $\forall i \in \{1, 2, ..., m\} : |q_i - d_i| \leqq |q_i - d'_i|$, and at least one $j \in \{1, 2, ..., m\} : |q_j - d_j| < |q_j - d'_j|$.*

DSQ can be computed by using the traditional skyline computation(注1) by transforming all points $d^i = (d^i_1, d^i_2, ..., d^i_m)$ to the new data space w.r.t. point q, i.e., $d'^i = (|d^i_1 - q_1|, |d^i_2 - q_2|, ..., |d^i_m - q_m|)$. However, the query points registered in the system can be more than 1 point, e.g., many reference points. Let $Q = \{q^1, q^2, ..., q^{|Q|}\}$ be a set of monitoring dynamic skyline query points. In this case, Dynamic Skyline Query of Q is defined by $DSQ(D, Q) = \bigcup_{q \in Q} DSQ(D, q)$. In Fig.1a, $DSQ(D, q^1) = \{d^1, d^4, d^5\}$ because the traditional skyline on the new space consists of d'^1, d^4 and d'^5.

An event tuple e is a group of values read from different types of sensors associated with meta-data, e.g., a timestamp and a location. We assume that to acquire each attribute value costs differently, so cloud providers use a fixed price vector $C = \langle c^1, c^2, \ldots, c^m \rangle$ as a price per reading of attribute $att^1, att^2, \ldots, att^m$ respectively.

Definition 2. *(Dynamic Skyline Monitoring Query): Given a set of query points $Q \subset A$ and a dataset D, Dynamic Skyline Monitoring Query (DSMQ) according to Q identifies whether an m-dimensional event tuple $e = (e_1, e_2, ..., e_m)$ belong to a dynamic skyline w.r.t. at least one query $q \in Q$.*

$$DSMQ(D, Q, e) = \begin{cases} true & ; e \in DSQ(D \cup \{e\}, Q) \\ false & ; Otherwise \end{cases}$$

In Fig.1b, event tuple e^1 is dynamically dominated by d^4 while e^2 is not dynamically dominated by any point, so $DSMQ(D, \{q^1\}, e^1)$ returns $false$ and $DSMQ(D, \{q^1\}, e^2)$ returns $true$.

(注1) : The definition of traditional skyline is referred to [2], [3], [7]

To periodically fetch all attributes for monitoring applications, the cost is fixed at $\sum_{i=1}^{m} c^i$ at each timestamp. In this paper, we object to execute monitoring DSMQ by minimizing the total expense for Sensor-Cloud services. We observe that, many monitoring applications only focus on unusual events (returning false in DSMQ) in real-time while the others (possibly a majority) are neglected not even to be stored into the disk. Note that if users prefer to access historical data to analyze in details, they may pay additional fees to access those data directly from the cloud.

3.2 Sequential Attribute Request Strategy

Instead of getting all attribute values, it is more economical to sequentially get only a set of partial attributes that is still able to lead to the same decision making. However, that optimal set is unknown in prior. Our proposed framework presents the strategy to sequentially request some promising attributes that are still able to answer DSMQ.

Here, we give the definition of anti-dominance range of attribute att_i.

Definition 3. *(Anti-dominance range of attribute att_i w.r.t q denoted by $AD_q(att_i)$): Given a dataset D and a query point q, $AD_q(att_i)$ is an interval $[a, b] \in [0, A_i]$ which is expressed by:*

$$AD_q(att_i) = [\max(0, q_i - g), \min(A_i, q_i + g)] \qquad (1)$$

where $g = \min_{s \in q.DSQ} |s_i - q_i|$

Lemma 1. *For any $e_i \in AD_q(att_i)$ regardless of other attribute values then $e \in DSQ(D \cup \{e\}, q)$*

Proof. Omitted. □

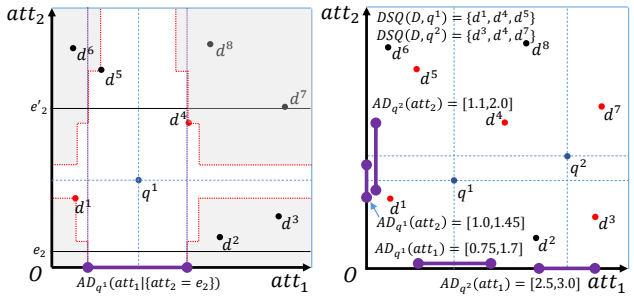
The examples of $AD_{q^1}(att_1)$ and $AD_{q^1}(att_2)$ (the purple intervals) are shown in Fig.1b. Note that because $e_1^2 \in AD_{q^1}(att_1)$ regardless of e_2^2 then $DSMQ(D, Q, e^2) = true$.

However, at the beginning, we have to decide which attribute to be retrieved first. The greedy choice is to retrieve the attribute which is most likely able to draw the conclusion right after the request. Therefore, assuming that all attributes are independent and identically distributed random variables (iid), we can calculate the probability of $DSMQ(D, Q, e) = true$ (Dynamic skyline probability) after having requested att_i (e_i) by the following equation.

$$P_{D,Q}(att_i) = \frac{len(\bigcup_{q \in Q} AD_q(att_i))}{A_i} \qquad (2)$$

where $len()$ is a sum of lengths of union intervals.

In the case that after having fetched the first attribute and $DSMQ(D, Q, e)$ cannot be concluded, we have to choose the next promising attribute to be requested from the remaining unrequested attributes. Given the already-fetched attribute



(a) Anti-dominance ranges after fetching e_2 (b) Running example

Figure 2: An example of anti-dominance range

set F, then $e_i : att_i \in F$ is known, the anti-dominance range and skyline probability of $att_i \in AT \backslash F$ are possibly changed as the following equations.

$$AD_q(att_i|F) = [\max(0, q_i - g'), \min(A_i, q_i + g')] \qquad (3)$$

where $g' = \min_{s \in q.DSQ \backslash \{s' \in q.DSQ | \forall j \in F : |q_j - s_j| > e_j\}} |s_i - q_i|$.

In Fig.2a using the same dataset as Fig.1b, assume that e_2 has been fetched, the range $AD_{q^1}(att_1|\{e_2\})$ is expanded from $AD_{q^1}(att_1)$ as shown in the figure.

$$P_{D,Q}(att_i|F) = \frac{len(\bigcup_{q \in Q} AD_q(att_i|F))}{A_i} \qquad (4)$$

Lemma 2. *If $F' \subseteqq F$, then $P_{D,Q}(att_i|F') \leqq P_{D,Q}(att_i|F)$*

Proof. Omitted. □

Nevertheless, the important factor is cost. Our proposed framework selects a promising attribute to fetch based on the following cost-efficient measurement.

$$att_i = \underset{att_i \in AT \backslash F}{\arg\max} \left[\frac{P_{D,Q}(att_i|F)}{c_i} \right] \qquad (5)$$

Running Example : We illustrate by using Fig.2b. Given a price vector is $\langle 3, 2 \rangle$, we calculate the probability as follows: $P_{D,Q}(att_1) = 1.4/3 = 0.467$ and $P_{D,Q}(att_2) = 1.0/2 = 0.5$. Hence, The first attribute to select is att_2.

Algorithm 1 summarizes the procedures of our proposed framework. We assume that $DSQ(D, q)$ for $q \in Q$ can be pre-computed once.

3.2.1 Time Complexity Analysis

Finding $AD_q(att_i|F)$ requires calculating g' for each q and each attribute. This takes $O(m \cdot |Q| \cdot |S|)$ time where $|S|$ is the cardinality of the dynamic skyline. For independent dimensions, the expected number of skyline points is $\theta((\log n)^d/(d-1)!)$ [2]. In line 4, $len(\bigcup_{q \in Q} AD_q(att_i|F))$ can be found by sorting intervals and computing their sum with $O(|Q| \log |Q|)$ time. In line 8, checking the membership is processed on an interval tree in $O(\log |Q|)$ time. For each tuple, in the worst case, these procedures need to be repeated

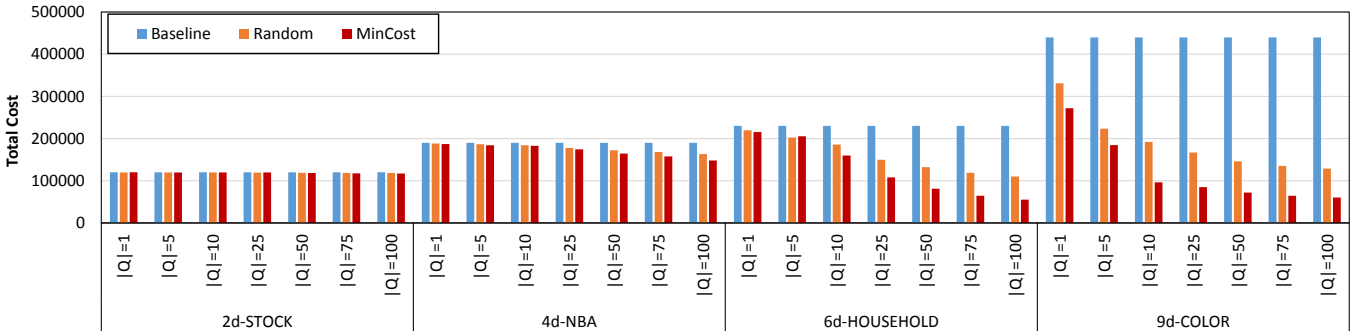


Figure 3: Total Cost (units)

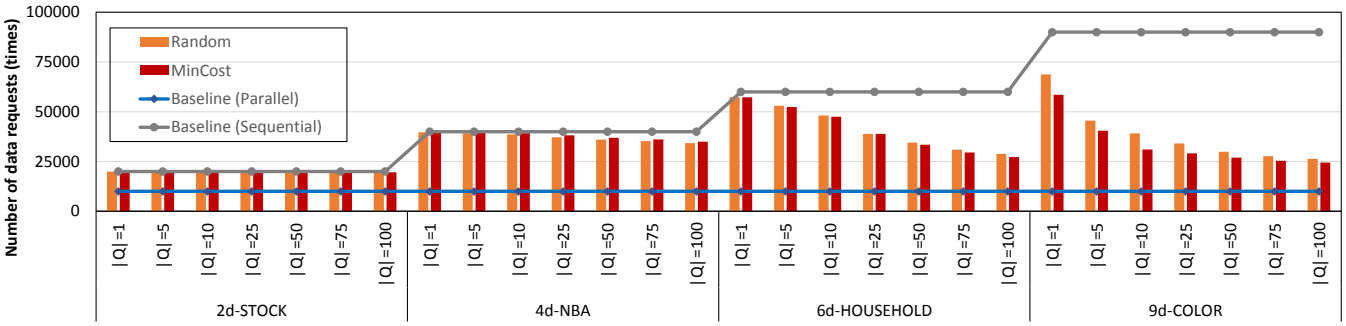


Figure 4: The number of data requests (times)

Algorithm 1: The proposed framework

Data: a dataset D, a query set Q, an event tuple e, a price vector c

Result: $DSMQ(D, Q, e)$, $cost$

1 $F \leftarrow \emptyset$
2 **while** $F \nmid Q$ **do**
3 Calculate $\bigcup_{q \in Q} AD_q(att_i|F)$
4 Calculate $P_{D,Q}(att_i|F)$ for all $att_i \in AT \backslash F$
5 Request e_i where $att_i = \arg\max_{att_i \in AT \backslash F} P_{D,Q}(att_i|F)/c_i$
6 $cost \leftarrow cost + c_i$
7 $F \leftarrow F \cup \{att_i\}$
8 **if** $e_i \in \bigcup_{q \in Q} AD_q(att_i)$ **then**
9 return (true, $cost$)
10 return (false, $cost$)

at most m times resulting in $O(m^2 \cdot |Q| \cdot |S| + m \cdot |Q| \log |Q|)$ in total.

4. Experiments and Discussion

We empirically evaluate our proposed framework's efficiency. All algorithms were implemented in C# and performed in a commodity PC (8GB, Core i7 3.6GHz). Since none of existing studies addressed this problem before, we compared our framework with (i) Baseline – fetching all attributes (ii) Random – choosing next attribute randomly by replacing lines 4-5, Algorithm 1. We used various dimensional real data sets including STOCK (2d) (from NYSE), NBA (4d), HOUSEHOLD (6d) and COLOR (9d) used in [5], [10]. We sampled 1000 points from each data set as a dataset D ($N_D = 1000$). The set of query points (Q) was drawn from independent data distribution varying between 1-100 points. The actual event tuples are also sampled from its corresponding dataset. Price vectors specified in each dataset were randomly generated with a random integer price between 1-9. As a result, $\langle 4, 8 \rangle$, $\langle 7, 1, 9, 2 \rangle$, $\langle 1, 9, 3, 4, 2, 4 \rangle$ and $\langle 2, 2, 5, 4, 5, 8, 4, 8, 6 \rangle$ were used as price vectors for 2d, 4d, 6d and 9d datasets respectively. Finally we measured the total cost of payment (10000 event tuples) as well as the total number of data requests as performance metrics.

In Fig.3, we show the total cost on different datasets and on various sizes of Q. The results indicat that the total cost incurred by our proposed framework (MinCost) is lower than the random scheme (Random) and the baseline (Baseline). The reduction ratio is getting higher on high $|Q|$ and high dimensionality (m) because the higher $|Q|$, the higher chance that an event tuple belongs to the dynamic skyline of one of the queries. In the same way, in high dimensionality, the size of dynamic skyline of each query ($|DSQ(D, q^i)|$) is supposed to increase exponentially. In low dimensionality i.e., 2d-STOCK, MinCost did not beat the other methods significantly (saving only 2.54% compared to the baseline when $|Q| = 100$). Nevertheless, the total cost can be saved up to 22.04%, 75.98% and 86.9% in 4d, 6d and 9d datasets respectively.

In Fig.4, we show the number of data requests on each setting. For the baseline, if we assume that we request all

attribute values once (parallel request), the number of requests is constant at 10000 (Baseline (Parallel)). Meanwhile, assuming sequential requests, we need to request $10000 \cdot m$ times (Baseline (Sequential)). The results also showed that the number of requests of MinCost is lower than the random scheme and the baseline in most cases especially on high dimensional datasets and high $|Q|$. However, it does not mean that requesting fewer attributes results in lower cost. To illustrate, in 4d-NBA dataset and $|Q| \in [10, 100]$, MinCost requested more attribute values than the random scheme, but MinCost consumes lower cost because MinCost makes a retrieval plan based on both skyline probability and cost of each attribute.

In our framework, finding $AD_{q^i}(att_i|F)$ for each q^i and att_i is the highest intensive computation. However, in our experiments even in 9d, $|Q| = 100$, this process takes only 42ms in the worst case for single event tuple, while less complicated cases and other process take less than 1ms. Therefore, our proposed framework does not burden high latency, and applicable to most real-time applications.

5. Conclusion and Future work

We have introduced the problem of dynamic skyline monitoring on Sensor-Cloud and proposed a novel approach to save the compensation for it. Our approach iteratively requests and assesses the most promising attribute first by measuring our proposed cost-effective attribute measurement. Hence some tuples can be identified as dynamic skyline without fetching all attributes. The experiments performed on various real datasets ensure that this strategy can significantly cut the cost compared to the baseline and outperform the random strategy. In addition, the processing cost is quite small even in high dimensionality and many query points.

As a part of our future work, we aim to investigate the same problem on other types of queries and the feasibility to employ learning models for adaptively fitting to an irregular data distribution.

6. Acknowledgements

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

References

[1] A. Alamri, W. S. Ansari, M. M. Hassan, M. S. Hossain, A. Alelaiwi, and M. A. Hossain. A survey on sensor-cloud: architecture, applications, and approaches. *IJDSN*, 2013.

[2] E. Dellis and B. Seeger. Efficient computation of reverse skyline queries. In *VLDB*, pages 291–302, 2007.

[3] K. Hose and A. Vlachou. A survey of skyline processing in highly distributed environments. *VLDB Journal*, 21(3):359–384, 2012.

[4] S. Madria, V. Kumar, and R. Dalvi. Sensor cloud: A cloud of virtual sensors. *Software, IEEE*, 31(2):70–77, Mar 2014.

[5] D. Nanongkai, A. D. Sarma, A. Lall, R. J. Lipton, and J. Xu. Regret-minimizing representative databases. *VLDB*, 3(1-2):1114–1124, 2010.

[6] D. Papadias, Y. Tao, G. Fu, and B. Seeger. An optimal and progressive algorithm for skyline queries. In *SIGMOD*, pages 467–478, 2003.

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

[8] X. Sheng, J. Tang, X. Xiao, and G. Xue. Sensing as a service: Challenges, solutions and future directions. *Sensors Journal, IEEE*, 13(10):3733–3741, 2013.

[9] D. Skoutas, D. Sacharidis, A. Simitsis, and T. Sellis. Serving the sky: Discovering and selecting semantic web services through dynamic skyline queries. In *ICSC*, pages 222–229, 2008.

[10] A. Sultana, N. Hassan, C. Li, J. Yang, and C. Yu. Incremental discovery of prominent situational facts. In *ICDE*, pages 112–123, 2014.