On Efficient Processing of Continuous Reverse Skyline Queries in Wireless Sensor Networks

Bo Yin[1,2]*, Siwang Zhou[2], Shiwen Zhang[3], Ke Gu[1], Fei Yu[1]
[1]Hunan Provincial Key Laboratory of Intelligent Processing of Big Data on Transportation, School of Computer and Communication Engineering, Changsha University of Science and technology, Changsha, 410004, China
[2] College of Information Science and Engineering, Hunan University, Changsha, 410082, China
[3]School of Computer Science and Engineering, Hunan University of Science and Technology, Xiangtan, 411201, China
[e-mail: yinbo@hnu.edu.cn, swzhou@hnu.edu.cn, shiwenzhang@hnu.edu.cn,gk4572@163.com,yufeiyfyf@163.com]
*Corresponding author: Bo Yin

Received September 20, 2016; revised December 14, 2016; accepted January 3, 2017; published April 30, 2017

Abstract

The reverse skyline query plays an important role in information searching applications. This paper deals with continuous reverse skyline queries in sensor networks, which retrieves reverse skylines as well as the set of nodes that reported them for continuous sampling epochs. Designing an energy-efficient approach to answer continuous reverse skyline queries is non-trivial because the reverse skyline query is not decomposable and a huge number of unqualified nodes need to report their sensor readings. In this paper, we develop a new algorithm that avoids transmission of updates from nodes that cannot influence the reverse skyline. We propose a data mapping scheme to estimate sensor readings and determine their dominance relationships without having to know the true values. We also theoretically analyze the properties for reverse skyline computation, and propose efficient pruning techniques while guaranteeing the correctness of the answer. An extensive experimental evaluation demonstrates the efficiency of our approach.

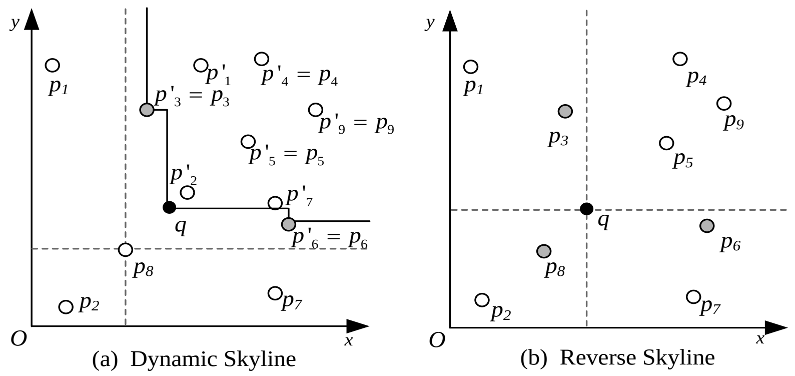
Keywords: Reverse skyline, continuous query, wireless sensor network, energy efficiency, pruning

This research is supported by the Natural Science Foundation of China (Grant No.61402055), the Natural Science Foundation of Hunan Province, China (Grant No.2016JJ3012), and the Open Research Fund of Hunan Provincial Key Laboratory of Intelligent Processing of Big Data on Transportation, China.

1. Introduction

A wireless sensor network (WSN) is a collection of sensor nodes, which are responsible for sensing the surrounding environment and collaborating with each other to relay the sensed data to a centralized location or sink to answer user queries. Conventional queries, such as top-k queries [1], and nearest neighbor queries [2], have yet been successfully adapted to WSN networks. The success of such queries and development of WSN hardware techniques undoubtedly encourage further attempts to adapt more complicated queries to the WSNs [3]. As an important query operator for intelligent decision over complex data, where multiple and often conflicting criteria are considered, the skyline operator [4] and its variants [5]-[16] have been extensively studied recently. In this paper, we study the problem of answering *continuous reverse skyline query in WSNs*, which seeks to find the reverse skylines as well as the full set of nodes that reported them for a number of continuous sampling epochs.

To explain reverse skyline, we explain dynamic skyline first. The dynamic skyline [5] considers “relative” values, i.e., the coordinate-wise distances between data points and a user-given query point, and returns all those points that are not dominated by any other point with respect to the query point. Specifically, a point p_1 dominates another p_2, if the distance is not larger than that of p_2 on every dimension, and smaller on at least one dimension. **Fig. 1(a)** shows an example of dynamic skyline of point p_8. Each point p=(p[1], p[2]) in original 2D space is transformed to a point p'=(|p[1]-p_8[1]|, |p[2]-p_8[2]|) in 2D distance space. The dynamic skyline of p_8 is {p_3, p_6, q}. The dynamic skyline query produces interesting points from the “user” perspective, that is, users who are interested in p_8 are likely to be interested in dynamic skyline points p_3, p_6 and q. Based on dynamic skyline, a reverse skyline query retrieves a set of points whose dynamic skyline contains the query point [6]. **Fig. 1(b)** shows the reverse skyline set of query point q. Since q is contianed in the dynamic skyline of p_8, p_8 is contained in the reverse skyline of q. The same holds for p_3 and p_6. Hence, the reverse skyline set of q is {p_3, p_6 , p_8}. The reverse skyline focuses on the “companies” perspective. It means that users in reverse skyline set would have been interested in the company product q. Continuous reverse skyline query is very useful in many WSN applications.



(a) Dynamic Skyline (b) Reverse Skyline

Fig. 1. Reverse skyline example

Scenario 1. For providing valuable services for gardeners (e.g., where and what to plant), sensors are deployed to monitor weather and soil conditions (temperature, humidity, light, etc.). The plant species are represented by their preferences on different environment parameters. The query point *q* is specified as particular monitoring areas. If a plant species is in the reverse skyline of *q*, this species would prefer the area that *q* represents. Then, plant species appear most in one-day reverse skyline query results is considered to be planted at *q*. Given some new locations, the results of reverse skyline querying can also be employed to seek the suitable area that more plant species are interested in.

Scenario 2. In underground coal mines, to ensure safe working conditions for miners, we may use sensor networks to collect data, such as gas density, oxygen density, and water depth. Let us treat the critical values of possible disasters as query *q*, and continuously pose reverse skyline query. If most of the query results in a period of time (e.g., 15 minutes) come from a particular node (or its neighbors), the located area is probably in dangerous state.

Challenge. Answering continuous reverse skyline query in WSNs is non-trivial. As the reverse skyline operator is not decomposable, even if a point is identified as not belonging to the query results, this point cannot be simply discarded. Otherwise, false-positive results can occur. Therefore, during the query processing, a great number of unqualified data points are transferred over the network for the final results, which incurs high communication cost.

Limitations of the State-of-the-art techniques. The reverse skyline query has been extensively studied ever since it was introduced in [6]. In addition to traditional reverse skyline query [6,7], a variety of query variants have been studied, such as why-not reverse skyline query [8], bichromatic reverse skyline query [9]-[12], ranked reverse skyline query [13], group-by reverse skyline[14], and reverse skyline over sliding windows [15,16], and different application environments are considered, such as non-metric space[17], mapreduce framework [18], uncertain database [19], and moving scenario [20]. Nevertheless, most existing studies focus on snapshot queries in centralized systems. Their approaches employ centralized data structures like R-tree and are optimized for processing cost.

The most related works are [16] and [21], which study reverse skyline query in WSNs. The approach in [21] is designed for snapshot queries and works on the idea of converting the reverse skyline problem into a full skyline problem such that in-network processing can be utilized to reduce the network traffic. This approach is extended to handle reverse skyline queries over sliding windows in [16]. Although the snapshot reverse skyline results could be extended to answer the reverse skyline query in one or several sampling epochs, it may incur significant overhead. This is because, every sensor node needs to report its sensor reading at each query time. Nevertheless, most of those sensor readings may not belong to the final results. Furthermore, data points are pruned based on routing information, not global information, which greatly decreases the pruning effectiveness.

Contributions. In this paper, we propose a new algorithm, called efficient computation of reverse skyline (ECRS), which avoids transmission of updates from nodes that cannot influence the reverse skyline. Our basic idea is to utilize mapping information of sensor readings to identify nodes that produced reverse skylines and prune nodes that cannot belong to the final results. We first propose a mapping scheme, which maps a sensor reading to a hypersquare that bounds the value of every attribute of the sensor reading. The hypersquare can be represented by its center and the deviation, where the center is a history sensor reading which is archived at both source node and the sink. Hence, in stead of reporting the true sensor readings, sensor nodes send the one-dimension deviations of the hypersquares to the sink. Based on archived history data and received deviations, the sink can reconstruct the hypersquares of sensor nodes. We then theoretically analyze the properties for reverse skyline computation, and propose efficient pruning techniques while guaranteeing the correctness of the answer. Sensor nodes that can be pruned are excluded from the query results. In order to minimize the number of nodes that need to report their sensor readings, the sink conducts a pull-based acquisition of sensor readings from nodes whose sensor readings are determined as part of the reverse skyline set, such that these new obtained sensor readings can be used to prune more nodes. In summary, we make the following contributions in this paper.

- We propose an efficient algorithm that computes continuous reverse skyline in sensor networks. In order to reduce the communication cost, our proposed approach uses mapping information to identify nodes that produced reverse skylines and suppress unnecessary data retrieval.
- We propose a data mapping scheme which maps sensor readings to hypersquares, such that we can determine the dominance relationships of the sensor readings without having to know their true values.
- We theoretically analyze the properties for reverse skyline computation. We propose pruning techniques to reduce the number of nodes that need to report their sensor readings while guaranteeing the correctness of the answer.
- Finally, we conduct extensive experimental study to evaluate the effectiveness of our algorithm.

The remaining part of this paper proceeds as follows: Section 2 reviews the related work and presents the preliminaries. Section 3 presents our proposed data mapping scheme. In section 4, we exploit the properties for reverse skyline computation based on mapped data. Section 5 presents the ECRS algorithm. Section 6 reports on experiments. Finally, we conclude in Section 7.

2. Related Work and Preliminaries

2.1 Related Work

The concept of reverse skyline is first introduced by Dellis and Seeger [6].The proposed algorithm firstly computes the global skyline points, a super set of the reverse skyline, and then determines reverse skyline points using window queries. Gao et al. [7] proposed a reuse mechanism to avoid multiple traversal of the R-tree to improve the performance of [6].

In addition to traditional reverse skyline query, a variety of query variants are studied as well. Islam et al. [8] addressed the problem of answering why-not questions in reverse skyline queries. Wu et al. [9] studied the bichromatic reverse skyline, and proposed several non-trivial heuristics to optimize the access order of R-tree to reduce I/O cost. Arvanitis et al. [10] proposed queue-based data structures to reduce processing cost of [9]. Lian et al.[11] considered bichromatic reverse skyline for uncertain datasets. Jiang et al. [12] studied the mutual reverse skyline queries and proposed and proposed several heap-based algorithms. Gao et al. [13] proposed data-partitioning index for ranked reverse skyline queries. Recently, the group-by reverse skyline query is studied in [14]. Works of [15] and [16] considered reverse skyline over sliding windows.

The reverse skyline queries under different application environments are also explored. Deshpande et al. [17] considered reverse skyline query with non-metirc similarity measures. The proposed algorithm utilized group-level reasoning and early pruning to reduce attribute level comparison. Park et al. [18] proposed efficient parallel algorithms for processing reverse skyline query using MapReduce. Bai et al. [19] proposed some probability pruning techniques for reverse skyline query over uncertain data stream. Lim et al. [20] considered moving objects, and proposed to make a verification range to reduce the search space and utilize the spatial index to improve the query efficiency.

Wang et al. [21] proposed a concept, called full skyline, as the set of candidate reverse skyline points. Since the full skyline query is decomposable, in-network processing can be utilized to reduce the network traffic during the query processing. Min et al.[16] extended the approach of [21] to handle reverse skyline queries over sliding windows. As discussed in Section 1, although the snapshot reverse skyline results [21] could be extended to answer the reverse skyline query in one or several time epochs, it may incur significant overhead. To answer continuous reverse skyline queries, we propose a query execution mechanism to avoid the updates from nodes not contributing to the final results. We propose a mapping scheme and pruning techniques to reduce the nodes that need to report their sensor readings. Since the pruning requirements are relaxed compared with that in [21] and extended for node pruning, our pruning techniques are much more efficient. Finally, [21] does not produce reverse skyline results progressively, in contrast to our approach.

There are some studies on skyline queries in WSNs. Chen et al. [22] considered continuous skyline computation, and used hierarchical thresholds to to reduce the transmission traffic. Kwon et al. [23] selected point closest to the origin as the filter, based on the idea that a point much closer to the origin has a higher pruning capability. Xin et al. [24] devised two types of filters, i.e., the grid filter and the tuple filter, for different data distributions. Liang et al. [25] proposed to use multiple points rather than a single point as the filter. The above methods assume that the sensor reading of each node is just stored locally. Differently, Su et al. [26] proposed a cluster-based architecture to store sensing readings, and proposed algorithms to avoid the need of collecting data from all nodes in the network. Different from these studies, in this paper we focuses on how to support continuose reverse skyline queries in WSNs.

2.2 Preliminaries

We assume a cluster-based sensor network [27], which extends the network lifetime and supports network scalability by grouping sensor nodes into clusters. Data collected by sensor nodes are first forwarded to corresponding cluster-heads. Then the cluster-heads route aggregated data of their clusters toward the sink to answer user queries. Each senor node s_i produces a d-dimensional point $p_i^t = (p_i^t[1], p_i^t[2], \cdots, p_i^t[d])$ at sampling epoch t. We omit the time epoch when it is clear from the context.

Definition 1 (Continuous reverse skyline query in WSN). The continuous reverse skyline query retrieves the set of reverse skyline points RSK^t with respect to query point q, for each sampling epoch t, as well as the set of nodes S^t that generated the reverse skyline points.

Definition 2 (Reverse Skyline [6]). Given a dataset D and a query point q, the reverse skyline with respect to q is a set of points whose dynamic skyline includes q. That is, a point $o \epsilon D$ is a reverse skyline point of q, iff $\nexists p \epsilon D$ such that (1) $|p[m]-o[m]| \leq |q[m]-o[m]|$ for all m, and (2) $|p[k]-o[k]| < |q[k]-o[k]|$ for at least one k.

3. Mapping Data into ε-hypersquares

The purpose of data mapping is to obtain approximate views of sensor readings, such that we can identify those nodes that produced reverse skyline points, and perform node pruning without having to know the true observation values. Due to spatio-temporal correlation among sensor readings, we propose to represent the approximate views based on history data. Note that, we archive a history sensor reading for each node at both the sink and source node in our

ECRS approach. Let p_i^r and p_i^t be the archived history and the current sensor readings of node s_i, respectively. We map p_i^t to an ε-hypersquare F_i^t centering at p_i^r with side length $2\varepsilon_i^t$. Since s_i and sink maintain the same p_i^r, s_i only sends ε_i^t to the sink which then reconstructs F_i^t as the approximate view of p_i^t. We call ε_i^t the deviation of p_i^t. Obviously, F_i^t is the minimum bounding hypersquare that centers at p_i^r and covers p_i^t. The maintainance and updation of history point p_i^r will be detailed in Section 5.

Definition 3 (ε-hypersquare). Given two data points p_i^t and p_i^r, the ε-hypersquare of p_i^t, denoted as F_i^t, is the hypersquare centering at p_i^r with side length $2\varepsilon_i^t$, where $\varepsilon_i^t = \max_{1\le m\le d} |p_i^r[m] - p_i^t[m]|$.

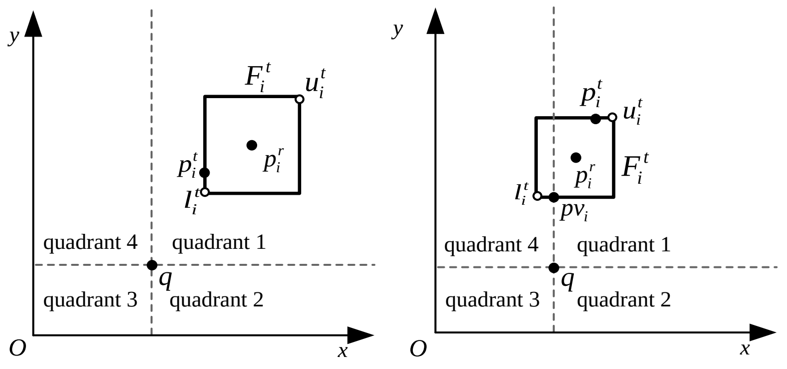
An ε-hypersquare F_i^t is represented by two corner points, i.e., the nearest corner l_i^t and the furthest corner u_i^t (to the query q). By conducting dominance tests among those corner points, we will justify whether or not the true observations are reverse skylines. We now formulate the computation of corner points. As shown in **Fig. 2**, the data space is cut into 2^d quadrants with d orthogonal hyper-planes. Note that whether point p is a reverse skyline point only depends on data points in the same quadrant. Points on quadrant intersection planes belong to all intersection quadrants. In order to get located quadrant of p_i^t based on p_i^r, we assume that p_i^t and p_i^r are in the same quadrant and p_i^r is not on quadrant intersection planes. When F_i^t is inside only one quadrant (**Fig. 2(a)**), corner points are calculated with Eq. (1) and Eq. (2).

$$u_i^t[m] = \begin{cases} p_i^r[m] + \varepsilon_i^t \mid q[m] < p_i^r[m] \\ p_i^r[m] - \varepsilon_i^t \mid q[m] > p_i^r[m] \end{cases}, 1 \le m \le d. \tag{1}$$

$$l_i^t[m] = \begin{cases} p_i^r[m] - \varepsilon_i^t \mid q[m] < p_i^r[m] \\ p_i^r[m] + \varepsilon_i^t \mid q[m] > p_i^r[m] \end{cases}, 1 \le m \le d. \tag{2}$$

Otherwise, we only need the part of F_i^t in the quadrant that p_i^r lies in. For this purpose, as illustrated in **Fig. 2(b)**, we treat point pv_i, which is on the quadrant intersection planes and nearest to q, as the new l_i^t as follows:

$$pv_i[m] = \begin{cases} \max(l_i^t[m], q[m]) \mid q[m] < p_i^r[m] \\ \min(l_i^t[m], q[m]) \mid q[m] > p_i^r[m] \end{cases}, 1 \le m \le d. \tag{3}$$



(a) An ε -hypersquare in one quadrant (b) An ε -hypersquare in multiple quadrants

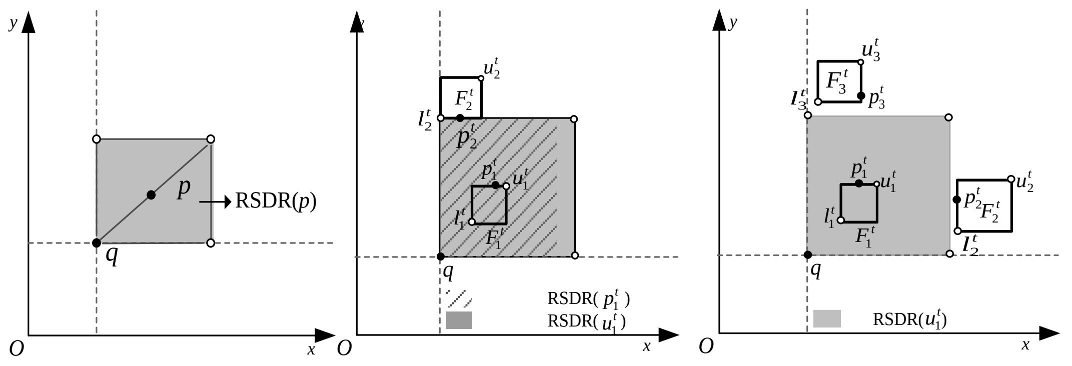
Fig. 2. Example of data mapping

4. Properties for Reverse Skyline Computation Based on Mapped Data

There are three aspects in reverse skyline calculation. These are: (1) identifying points that belong to reverse skyline query results; (2) identifying points that cannot belong to query results; and (3) safely pruning points not contributing to the final results. Recall that, in our mapping scheme, data points (i.e., the sensor readings) are represented with ε-hypersquares. Hence, we need to coduct the above three aspects based on mapped data. Given a sensor node represented with an ε-hypersquare, if its true data point can be pruned, this node need not to report its data point and it is immediately discarded. Furthermore, if the data point is determined as part of the reverse skyline set, we will conduct a pull-based acquisition of the true observations from the nodes, and these observations will be used to prune more unqualified nodes. Hence, in the following illustration of properties for reverse skyline computation, we consider a mixed set of sensor readings and ε-hypersquares.

4.1 Identifying Reverse Skyline Points

Given a point p, the *reverse skyline dominance region* of p, denoted as RSDR(p), is the set of points dynamically dominating q with respect to p. Clearly, $p \in RSK(q)$ iff $\nexists o \in$ RSDR(p). To illustrate, consider the rectangle with q as a corner point and centered at p in **Fig. 3(a)**. We emphasize that RSDR(p) is this hyper-rectangle except corners points. Points in RSDR region can be justified based on the “semidominance” concept [21], that is, $o \in$ RSDR(p) if and only if o semidominates p. Specifically, point o semidominates p with respect to q, if: (1) $|o[m]-q[m]| \le 2|p[m]-q[m]|$ and $(o[m]-q[m])(p[m]-q[m]) \ge 0$ for all m, and (2) $|o[k]-q[k]| < 2|p[k]-q[k]|$ and $(o[k]-q[k])(p[k]-q[k]) > 0$ for at least one k.



(a) RSDR region (b) Comparisons based on RSDR regions (c) Identification of reverse skyline points

Fig. 3. Example of reverse skyline point identification

When data points are mapped into ε-hypersquares, intuitively, given an ε-hypersquare $F_i^t(l_i^t, u_i^t)$, if there does not exist another $F_h^t(l_h^t, u_h^t)$ such that l_h^t falls in RSDR(u_i^t), then p_h^t cannot be in RSDR(p_i^t). Nevertheless, it is not always true. For example, in **Fig. 3(b)**, l_2^t does not lie in RSDR(u_1^t), but its true observation p_2^t is in RSDR(p_1^t). This is because l_2^t is a corner of the rectangle with q as a corner point and centered at u_1^t. As stated earlier, RSDR(u_1^t) does not contain the corners, and thus, l_2^t does not belong to RSDR(u_1^t). To solve the problem, we define the extended semidominance (e-dominance) and prove two propositions (Proposition 1 and Proposition 2) that help to identify reverse skyline points from mapped data.

Definition 4 (Extended semidominance)*:* A point p_1 extendly semidominates (e-dominates) another point p_2 with respect to q, if (1) $|p_1[m]-q[m]| \le 2|p_2[m]-q[m]|$ and $(p_1[m]-q[m])(p_2[m]-q[m]) \ge 0$ for all m, and (2) $|p_1[k]-q[k]| < 2|p_2[k]-q[k]|$ and $(p_1[k]-q[k])(p_2[k]-q[k]) > 0$, or $p_1[k]-q[k]=0$, for at least one k.

Proposition 1. Point p_i^t is a reverse skyline point, if $\nexists$ p_j^t such that p_j^t semidominates p_i^t and $\nexists$ $F_h^t(l_h^t, u_h^t)$ such that l_h^t e-dominates p_i^t.

Proof. We prove the proposition using reduction to absurdity. Suppose that $P_i^t \notin RSK^t$, then p_h^t semidominates p_i^t. Hence, $\forall m \in d$, $|p_h^t[m]-q[m]| \le 2|p_i^t[m]-q[m]|$ and $(p_i^t[m]-q[m])(p_h^t[m]-q[m]) \ge 0$, and $\exists k \in d$, $|p_h^t[k]-q[k]| < 2|p_i^t[k]-q[k]|$ and $(p_i^t[k]-q[k])(p_h^t[k]-q[k]) > 0$. For the nearest corner l_h^t, it holds that $\forall m \in d$, $|l_h^t[m]-q[m]| \le |p_h^t[m]-q[m]|$ and $(l_h^t[m]-q[m])(p_h^t[m]-q[m]) \ge 0$. There are two conditions for l_h^t:

1. If l_h^t is on the quadrant intersection planes, i.e., $\exists k \in d$, $l_h^t[k]-q[k]=0$, we can get $\forall m \in d$, $|l_h^t[m]-q[m]| \le 2|p_i^t[m]-q[m]|$ and $(l_h^t[m]-q[m])(p_i^t[m]-q[m]) \ge 0$.

2. Otherwise, it holds that: $\forall m \in d$, $l_h^t[m]-q[m] \ne 0$. Then, we can infer (1) $\forall m \in d$, $|l_h^t[m]-q[m]| \le 2|p_i^t[m]-q[m]|$ and $(l_h^t[m]-q[m])(p_i^t[m]-q[m]) \ge 0$, and (2) $\exists k \in d$, $|l_h^t[k]-q[k]| < 2|p_i^t[k]-q[k]|$ and $(l_h^t[k]-q[k])(p_i^t[k]-q[k]) > 0$.

It follows from Definition 4 that l_h^t e-dominates p_i^t.

Proposition 2. Given an ε-hypersquare $F_i^t(l_i^t, u_i^t)$, its true observation p_i^t is a reverse skyline point, if $\nexists$ p_j^t such that p_j^t semidominates u_i^t and $\nexists$ $F_h^t(l_h^t, u_h^t)$ such that l_h^t *e*-dominates u_i^t.

Proof. We prove the proposition using reduction to absurdity. Suppose that $P_i^t \notin RSK^t$, then p_j^t or p_h^t semidominates p_i^t. If p_j^t semidominates p_i^t, we have $\forall m \in d$, $|p_j^t[m]-q[m]| \le 2|p_i^t[m]-q[m]|$ and $(p_i^t[m]-q[m])(p_j^t[m]-q[m]) \ge 0$, and $\exists k \in d$, $|p_j^t[k]-q[k]| < 2|p_i^t[k]-q[k]|$ and $(p_i^t[k]-q[k])(p_j^t[k]-q[k]) > 0$. It holds that $\forall m \in d$, $|p_i^t[m]-q[m]| \le |u_i^t[m]-q[m]|$ and $(p_i^t[m]-q[m])(u_i^t[m]-q[m]) \ge 0$. We can infer (1) $\forall m \in d$, $|p_j^t[m]-q[m]| \le 2|u_i^t[m]-q[m]|$ and $(p_j^t[m]-q[m])(u_i^t[m]-q[m]) \ge 0$, and (2) $\exists k \in d$, $|p_j^t[k]-q[k]| < 2|u_i^t[k]-q[k]|$ and $(p_j^t[k]-q[k])(u_i^t[k]-q[k]) > 0$, that is, p_j^t semidominates u_i^t. Hence, p_h^t semidominates u_i^t if p_h^t semidominates p_i^t. Thus, l_h^t *e*-dominates u_i^t according to Proposition 1.

From the above two propositions, we note that, if l_h^t does not *e*-dominate u_i^t, p_h^t cannot semidominate p_i^t, that is, $p_h^t \notin$ RSDR(p_i^t). **Fig. 3(c)** shows three ε-hypersquares and their true observations. Consider the ε-hypersquare $F_1^t(l_1^t, u_1^t)$. Since $|l_3^t[y]-q[y]| > 2|u_1^t[y]-q[y]|$, l_3^t does not *e*-dominate u_1^t. Similarily, l_2^t does not *e*-dominate u_1^t as $|l_2^t[x]-q[x]| > 2|u_1^t[x]-q[x]|$. Therefore, the true observation p_1^t is a reverse skyline point (Proposition 2).

4.2 Identifying Non-Reverse Skyline Points

Based on RSDR region, it is obvious that a point *p* cannot be a reverse skyline point, iff there is another point *o* in RSDR(*p*). That is, *o* semidominates *p*. We now extend the identification techniques of non-reverse skyline points to ε-hypersquares. Given two ε-hypersquares $F_i^t(l_i^t, u_i^t)$ and $F_j^t(l_j^t, u_j^t)$, if u_i^t falls in RSDR(l_j^t), it is likely that the true observation p_i^t is located in RSDR(p_j^t). Unfortunately, it is not always the truth. As shown in **Fig. 4**, although u_1^t is in RSDR(l_2^t), p_1^t is not inside RSDR(p_2^t) because p_1^t is located at the corner of the hypersquare that RSDR(p_2^t) belongs to. That is, p_1^t does not gurantee to semidominate p_2^t if u_1^t semidominates l_2^t. To solve this problem, we define strong semidominance (*s*-dominance) and prove two propositions (Proposition 3 and Proposition 4) that help identify non-reverse skyline points from mapped data.

Definition 5 (**Strong semidominance**). A point p_1 strongly semidominates (*s*-dominates) p_2 with respect to *q*, iff $|p_1[m]-q[m]| < 2|p_2[m]-q[m]|$ and $(p_1[m]-q[m])(p_2[m]-q[m]) \ge 0$ for all *m*.

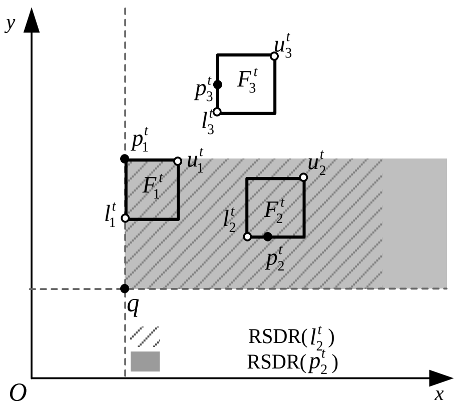


Fig. 4. Example of non-reverse skyline point identification

Proposition 3. Point p_i^t cannot be a reverse skyline point, if $\exists\ p_j^t$ such that p_j^t semidominates p_i^t or $\exists\ F_h^t(l_h^t, u_h^t)$ such that u_h^t s-dominates p_i^t.

Proof. According to Definition 5, since u_h^t s-dominates p_i^t, then $\forall m \in d$, $|u_h^t[m] - q[m]| < 2\,|p_i^t[m] - q[m]|$ and $(u_h^t[m] - q[m])(p_i^t[m] - q[m]) \geq 0$. Moreover, we have $\forall m \in d$, $|p_h^t[m] - q[m]| \leq |u_h^t[m] - q[m]|$ and $(p_h^t[m] - q[m])(u_h^t[m] - q[m]) \geq 0$. We can infer $\forall m \in d$, $|p_h^t[m] - q[m]| < 2\,|p_i^t[m] - q[m]|$ and $(p_h^t[m] - q[m])(p_i^t[m] - q[m]) \geq 0$. Since $\exists k \in d$ such that $P_h^t[k] \neq q[k]$, we have $(P_h^t[k] - q[k])(p_i^t[k] - q[k]) > 0$. Hence, p_h^t semidominates p_i^t if u_h^t s-dominates p_i^t.

Proposition 4. The true observation of $F_i^t(l_i^t, u_i^t)$ cannot be a reverse skyline point, if $\exists\ p_j^t$ such that p_j^t semidominates l_i^t or $\exists\ F_h^t(l_h^t, u_h^t)$ such that u_h^t s-dominates l_i^t.

Proof. If p_j^t semidominates l_i^t, we can get $\forall m \in d$, $|p_j^t[m] - q[m]| \leq 2\,|l_i^t[m] - q[m]|$ and $(p_j^t[m] - q[m])(l_i^t[m] - q[m]) \geq 0$, and $\exists k \in d$, $|p_j^t[k] - q[k]| < 2\,|l_i^t[k] - q[k]|$ and $(p_j^t[k] - q[k])(l_i^t[k] - q[k]) > 0$. It holds that $\forall m \in d$, $|l_i^t[m] - q[m]| \leq |p_i^t[m] - q[m]|$ and $(l_i^t[m] - q[m])(p_i^t[m] - q[m]) \geq 0$. We can infer (1) $\forall m \in d$, $|p_j^t[m] - q[m]| \leq 2\,|p_i^t[m] - q[m]|$ and $(p_j^t[m] - q[m])(p_i^t[m] - q[m]) \geq 0$, and (2) $\exists k \in d$, $|p_j^t[k] - q[k]| < 2\,|p_i^t[k] - q[k]|$ and $(p_j^t[k] - q[k])(p_i^t[k] - q[k]) > 0$. That is, p_j^t semidominates p_i^t. Then, if u_h^t s-dominates l_i^t, we can infer p_h^t semidominates l_i^t according to Proposition 3, and thus, p_h^t semidominates p_i^t.

In the example of **Fig. 4**, u_1^t s-dominates l_3^t because $|u_1^t[m] - q[m]| < 2\,|l_3^t[m] - q[m]|$ on all dimensions. According to Proposition 4, the true observation p_1^t must semidominate p_3^t. Therefore, p_3^t cannot belong to reverse skyline set.

4.3 Safe Pruning

In this subsection, we develop new pruning strategies for the reverse skyline queries while returning the correct query answer set.

Given a point *p*, let *S*(*P*) denote the set of data points that *p* semidominates. Data set *F*(*p*) contains points that fully dominates *p* [21]. Specifically, point p_1 fully dominates another p_2 with respect to *q* if: (1) $|p_1[m]-q[m]|\le|p_2[m]-q[m]|$ and $(p_1[m]-q[m])(p_2[m]-q[m])\ge 0$ for all *m*, and (2) $|p_1[k]-q[k]|<|p_2[k]-q[k]|$ and $(p_1[k]-q[k])(p_2[k]-q[k])>0$ for at least one *k*. We denote the region formed by points in *F*(*p*) as FDR(*p*), i.e., the rectangle with *p* and the query point *q* as diagonal corners but except rectangle corners, as shown in **Fig. 5**. Let us divide region FDR(*p*) into two regions $\text{FDR}_1(p)$ and $\text{FDR}_2(p)$. Specifically, $\text{FDR}_1(p)$ contains data points in $F(p)\cap S(p)$. In the work of [21], point *p* can be pruned only if there are at least two points in region FDR(*p*). We prove one lemma (Lemma 1) to improve the pruning efficiency. Note that, point *o* must semidominate *p* if *o* fully dominates *p*.

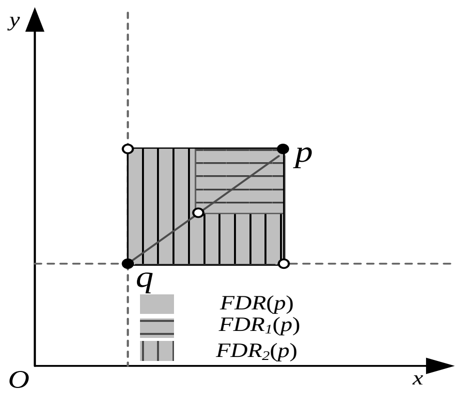


Fig. 5. FDR region

Lemma 1. Point *p* can be pruned, if there are at least (1) two points in region $\text{FDR}_1(p)$, or (2) one point in region $\text{FDR}_2(p)$.

Proof. Point *p* can be pruned, iff (1) *p* is a non-reverse skyline point, and (2) any point in *S*(*p*) is semidominated by another point except *p*. The first condition is fulfilled because *p* falls in region FDR(*p*). For any point $o\epsilon S(p)$, points in region FDR(*p*) semidominate *o*. Since points in $\text{FDR}_1(p)$ also belong to *S*(*p*), it is equivalent to proving that the two points in $\text{FDR}_1(p)$ semidominate each other. This is straightforward because points in $\text{FDR}_1(p)$ fully dominates *p* and *p* semidomiates those points.

We extend Lemma 1 to *ε*-hypersquares based on the notion of strong fulldominance (*f*-dominance). Given a point p_i^t, we say that $F_h^t \in \text{FDR}(p_i^t)$ if u_h^t *f*-dominates p_i^t, and $F_h^t \in \text{FDR}_1(p_i^t)$ if another condition is fulfilled, i.e., p_i^t semidominates l_h^t. It guarantees that the true observation p_h^t really falls in $\text{FDR}(p_i^t)$ or $\text{FDR}_1(p_i^t)$ (Proposition 5).

Similarly, given an *ε*-hypersquare $F_i^t(l_i^t,u_i^t)$, we say that point $p_j^t\epsilon\text{FDR}(F_i^t)$ if p_j^t fully dominates l_i^t, and $p_j^t\epsilon\text{FDR}_1(F_i^t)$ if u_i^t *s*-dominates p_j^t and p_j^t fully dominates l_i^t. The *ε*-hypersquare $F_h^t(l_h^t,u_h^t)\epsilon\text{FDR}(F_i^t)$ if u_h^t *f*-dominates l_i^t, and $F_h^t(l_h^t,u_h^t)\epsilon\text{FDR}_1(F_i^t)$ if another condition is fulfilled, i.e., u_i^t *s*-dominates l_h^t.

Definition 6 (**Strong fulldominance**). A point p_1 strongly fulldominates (*f*-dominates) p_2 with respect to *q*, if $|p_1[m]-q[m]|<|p_2[m]-q[m]|$ and $(p_1[m]-q[m])(p_2[m]-q[m])\geq 0$ for all *m*.

Lemma 2. Given a point p_i^t and an *ε*-hypersquare $F_h^t(l_h^t,u_h^t)$, p_h^t fully dominates p_i^t if u_h^t *f*-dominates p_i^t.

Proof. Since u_h^t *f*-dominates p_i^t, we can get $\forall m\in d$, $|u_h^t[m]-q[m]|<|p_i^t[m]-q[m]|$ and $(u_h^t[m]-q[m])(p_i^t[m]-q[m])\geq 0$. It holds that $\forall m\in d$, $|p_h^t[m]-q[m]|\leq|u_h^t[m]-q[m]$ and $(p_h^t[m]-q[m])(u_h^t[m]-q[m])\geq 0$. We can infer $\forall m\in d$, $|p_h^t[m]-q[m]|<|p_i^t[m]-q[m]|$, and $(p_h^t[m]-q[m])(p_i^t[m]-q[m])\geq 0$. Since $\exists k\in d$, $p_h^t[k]\neq q[k]$, then $(p_h^t[k]-q[k])(p_i^t[k]-q[k])>0$. Thus, p_h^t fully dominates p_i^t.

Proposition 5. A point or an *ε*-hypersquare can be pruned, if there are at least (1) two points/*ε*-hypersquares in the FDR$_1$ region, or (2) one point/*ε*-hypersquare in FDR$_2$ region.

Proof. 1. Given a point p_i^t, if *ε*-hypersquare F_h^t ϵFDR(p_i^t), we can get u_h^t *f*-dominates p_i^t. According to Lemma 2, we can infer p_h^t fully dominates p_i^t. Hence, p_h^t ϵFDR(p_i^t). When F_h^t ϵFDR$_1$(p_i^t), we can get p_i^t semidominates l_h^t. Therefore, p_i^t semidominates p_h^t, i.e., p_h^t ϵFDR$_1$(p_i^t).

2. Given an *ε*-hypersquare $F_o^t(l_o^t,u_o^t)$, if point *p*ϵFDR(F_o^t), we can get *p* fully dominates l_o^t. Clearly, *p* fully dominates p_o^t, i.e., *p*ϵFDR(p_o^t). When *p*ϵFDR$_1$(F_o^t), we have u_o^t *s*-dominates *p*. We can infer that p_o^t semidominates *p* according to Proposition 3. Hence, *p*ϵFDR$_1$(p_o^t). Furthermore, if another *ε*-hypersquare F_h^t ϵFDR(F_o^t), we can get u_h^t *f*-dominates l_o^t. According to Lemma 2, we can infer p_h^t fully dominates l_o^t. Hence, p_h^t fully dominates p_o^t, i.e., p_h^t ϵFDR(p_o^t). When F_h^t ϵFDR$_1$(F_o^t), we have u_o^t *s*-dominates l_h^t. We can infer p_o^t semidominates l_h^t according to Proposition 3. Therefore, p_o^t semidominates p_h^t, i.e., p_h^t ϵFDR$_1$(p_o^t).

In conclusion, there are always at least two points in FDR$_1$ region, or one point in FDR$_2$ region. Hence, point p_i^t and *ε*-hypersquare F_o^t can be safely pruned based on Lemma 1.

Consider the *ε*-hypersquares in **Fig. 6**, since u_3^t *f*-dominates l_1^t and u_1^t *s*-dominates l_3^t, we can get F_3^t ϵFDR$_1$(F_1^t). Furthermore, u_2^t *f*-dominates l_1^t but u_1^t does not *s*-dominate l_2^t, and thus, F_2^t ϵFDR$_2$(F_1^t). Since there exists an *ε*-hypersquare in the FDR$_2$ region of F_1^t, F_1^t can be immediately discarded according to Proposition 5.

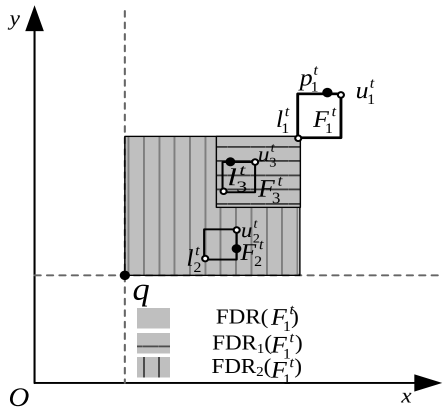


Fig. 6. Example of safe pruning

5. ECRS Algorithm

In this section we present our query processing algorithm ECRS. We first provide a general description in Section 5.1. We then go on to describe in detail the working of three query stages from Section 5.2 to Section 5.4.

5.1 Algorithm Framework

Initially, the sink collects sensor readings of all nodes and computes the initial reverse skyline query results. The sink also broadcasts into the network a user-specified threshold, referred to as ε, as the default value of deviation. Note that the sink maintains for each node the latest sensor reading that arrives at the sink and is not on quadrant intersection planes as the history data point. This sensor reading is also archived at both source node and corresponding cluster-head. Therefore, at each sampling epoch, both the sink and head node can get the same approximate views of the true observations, and then safely prune unqualified nodes while avoiding false-positives.

Thereafter, ECRS at each subsequent time epochs consists of three stages.

(1) Mapping information collection: the purpose of this stage is to gather mapped data of sensor readings, and then identify sensor nodes that produced the reverse skyline points and perform node pruning.

(2) Probing: the sink asks for sensor readings of “special” nodes (e.g., the identified nodes that produced the reverse skyline points), and performs node pruning once again based on these new obtained sensor readings.

(3) Complementing the final results: the sink pulls observations from nodes that have not reported their sensor readings and not pruned, and generates the rest reverse skylines.

Note that, the reverse skyline points can be obtained in all three stages. If a point is determined as a reverse skyline point, it can be immediately reported as part of the final results. Hence, the query results are returned progressively in our approach.

5.2 Mapping Information Collection

During this stage, sensor nodes trigger a propagation of mapping information. Since the user-specified threshold ε is set as the default value of deviation, node s_i reports to cluster head its deviation ε_i^t only when $\varepsilon_i^t > \varepsilon$ and the true observation p_i^t is not on quadrant intersection planes. For the scenario that p_i^t falls on the quadrant intersection planes, p_i^t is directly sent to cluster head.

After receiving the mapping information, cluster-heads compute ε-hypersquares and apply Proposition 5 to prune unqualified ε-hypersquares as well as data points. Clearly, sensor nodes whose ε-hypersquares are pruned need not report their data point in further processing, and thus, they are immediately discarded. We also discard the points or ε-hypersquares with only one point/ε-hypersquare in FDR_1 region and mark this point/ε-hypersquare as not belonging to the reverse skyline query results. We now illustrate the reason. For convenience, we consider point *p*, and let *o* be a point/ε-hypersquare in $FDR_1(p)$. Obviously *p* and point *o* (or the true observation of ε-hypersquare) semidominate each other and they cannot belong to final results. Since points of *S*(*p*) are also semidominated by *o*, those points can be still determined as not belonging to revere skyline without *p*. Hence, *p* can be safely pruned by marking *o*. When a point is discarded, downlink message is transmitted back to source node to inform it that the sensing reading has not arrived at sink and the archived history point need not to be updated. The purpose is to maintain the consistency among source node, cluster-head and the sink.

Then, cluster-heads refine the mapping information for unpruned data points. Especially, for each unpruned point p_i^t, cluster-head selects the history point p_l^s (from the archived history data of the cluster) with the minimum value of deviation and in the same quadrant of p_i^t, and sends the new deviation to the sink. The corresponding node information is also sent to the sink, such that the sink can get a correct view of F_i^t.

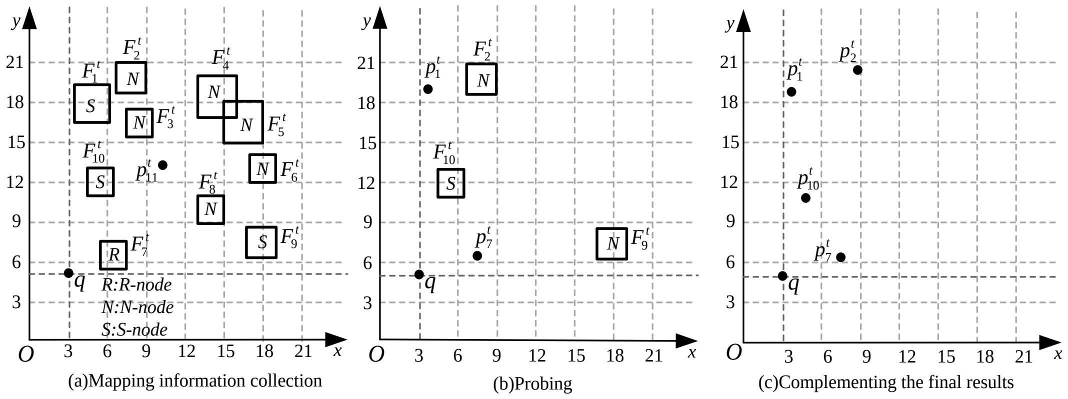

(a)Mapping information collection
R:R-node
N:N-node
S:S-node
(b)Probing
(c)Complementing the final results

Fig. 7. Example of ECRS

Based on obtained mapping information, the sink prunes away unqualified nodes and partitions the remaining nodes into three groups: the “reverse skyline” group (or R-node), the “non-reverse skyline” group (or N-node), and the group of nodes such that whether these nodes belong to the query result is still unknown (or S-node). Consider the example in Fig. 7(a) which shows ε-hypersquares of ten sensor nodes (s_1 to s_{10}) and one point p_{11}^t of node s_{11}. The threshold ε is set to 1. Node s_7 is a R-node, since no point semidominats u_7^t and there is no ε-hypersquare whose nearest corner *e*-dominates u_7^t (according to Proposition 2). The N-node set is $\{s_2, s_3, s_4, s_5, s_6, s_8, s_{11}\}$ according to Proposition 3 and Proposition 4. The ε-hypersquares

{ $F_3^t, F_4^t, F_5^t, F_6^t, F_8^t$ }and point p_{11}^t can be immediately discarded, since they have F_7^t located in their FDR_2 region (according to Proposition 5). The rest nodes {s_1 s_9, s_{10}} are S-nodes because we cannot determined whether they belong to the query results.

5.3 Probing

In the probing stage, the sink requires the sensor readings of all R-nodes, as these sensor readings belong to the reverse skyline set. The sink also asks some special S-nodes to report their true observations. These special S-nodes are part of S-nodes such that we can not determine whether they belong to the query results if we know the sensor readings of all other nodes. Thus, these nodes must report their true observations. Specifically, in order to identify special S-nodes, we make the following observation:

Observation 1. Given an S-node s_i, represented by ε-hypersquare $F_i^t(l_i^t, u_i^t)$, s_i must report its sensor reading if l_i^t is not semidominated by l_h^t of any other $F_h^t(l_h^t, u_h^t)$.

The reason is straightforward. If the true observation p_i^t is a reverse skyline point, s_i should definitely send its observation. On the contrary, if p_i^t is not a reverse skyline point, we need to identify this "status" by using Proposition 4. Since s_i is an S-node, only when the true observation p_h^t semidominates l_i^t, we can determine p_i^t is not a reverse skyline point without having to retrieve the true values of p_i^t. Clearly, l_h^t semidominates l_i^t if p_h^t semidominates l_i^t. Hence, in order to determine whether or not p_i^t is a reverse skyline points, we must retrieve its true values.

The "probing" message containing the ids of "probing" nodes are then sent by the sink to corresponding cluster-heads, which thereafter retrieve observations from corresponding nodes. The cluster-heads perform node pruning based on received sensor readings and archived ε-hypersquares, which is similar with the processing in the first stage. Then, cluster-heads send unpruned sensor readings to the sink, which thereafter prune away unqualified nodes and identify sensor nodes that belong to the query result as in the first stage.

In **Fig. 7(a)**, s_1 is a "special" S-node that needs to send its true observation, because the nearest corner cannot be semidominated by that of any other ε-hypersquare (according to Observation 1). Therefore, in the probing stage, the sink asks for values from node s_1 and the R-node s_7. **Fig. 7(b)** shows updated mapping information. Since p_7^t falls in FDR$_2$ region of F_9^t, s_9 is an N-node and F_9^t can be pruned. It is obvious that node s_1 is a reverse skyline node. After the probing stage, s_{10} is still an S-node.

5.4 Complementing the Final Results

In the last stage, the sink requires sensor readings of all unpruned nodes and generates the rest reverse skyline points. Node pruning is performed at each cluster-head. Based on received sensor readings, the sink can correctly compute the rest reverse skyline points. Note that the reverse skyline result includes the reverse skyline points retrieved in all three stages. Since some sensor readings arrive at the sink, the sink updates the history data of corresponding nodes. The same updating operation is performed at cluster-heads and source nodes. Lemma 3 verifies the correctness of ECRS's algorithm.

As shown in **Fig. 7(c)**, the sink asks for true values of unpruned ε-hypersquares { F_2^t, F_{10}^t }.

Point p_{10}^t is reported as a reverse skyline point, and p_2^t does not belong to the reverse skylines since it is semidominated by p_1^t, p_7^t and p_{10}^t .

Lemma 3. At any epoch $t \geq t_{first}$, ECRS produces a correct reverse skyline result where t_{first} is the first sampling epoch.

Proof. The proof is obvious since the algorithm at the first sampling epoch is essentially a centralized processing, and the correctness at subsequent epochs is guaranteed by Propositions 1 to 5.

6. Performance Evaluation

In this section, we conduct a simulation-based performance of our proposed algorithm that is implemented in C++. All experiments were conducted on a PC equipped with a 3GHz Dual Core AMD processor equipped with 2GB RAM.

6.1 Experimental Setup

We assume that *n* sensor nodes are evenly deployed in $\sqrt{n} \times \sqrt{n}$ units with the sink located at the center. The communication radius is $2\sqrt{n}$ units. We employ HEED [27] as the underlying cluster-based network architecture. We deploy three different datasets: uniform, correalted and clustered [21,28]. We first generate the initial datasets that follow the above distributions respectively. For uniform and correlated datasets, we then randomly select data point from the initial dataset for each sensor node. For the clustered dataset, each cluster-head picks randomly a point and generate 4 centroids that follow a Gaussian distribution on each axis with variance 0.025, and a mean equal to the corresponding coordinate of the centroid. Then, all associated nodes obtain points, the coordinates of which follow a Gaussian distribution on each axis with variance 0.005 around the cluster centroids. Values of data points on each dimension are normalized between 0 and 1. Data points at each node s_i is then modeled as $p_i^t[l] = \lambda_i p_i^{t-1}[l] + e_i$, where $e_i \sim N(0, 0.1)$ and $\lambda_i \sim N(1, 0.01)$. Every node is initialized with $\lambda_i = 1$ and $e_i = 0$. Assume that each *d*-dimensional point holds $4 \times d$ bytes, and both a deviation and a node identifier take 4 bytes each.

Table 1. Experimental parameters and values

Parameter	Values
Dimensions	2, 3, **4**,5
Number of nodes	6000, 7000, **8000**, 9000
ε	0.006, 0.008, **0.01**, 0.02, 0.03, 0.04 (uniform dataset) 0.002, 0.004, **0.005**, 0.01, 0.015, 0.02 (clustered and correlated datasets)

Our main metrics are: (i) the node reduction rate (NRR): the proportion of non-reverse skyline nodes that do not send their sensor readings to the number of non-reverse skyline nodes, and (ii) the communication cost, which counts the number of bytes of messages transmitted during query processing. We compare our ECRS approach with *Basic* and *Mapping* approach. The *Basic* approach is a continuous version of algorithm of [21], where at each sampling epoch all nodes report their sensor readings and in-network processing based on the notion of full skyline is utilized to reduce the amount of transferred data. The *Mapping* approach is almost the same to our proposed ECRS method, except that it extends the pruning technique of

[21] to mapped data, that is, data points or ε-hypersquares are pruned only when there are at least two points/ε-hypersquares in the FDR regions. The parameters of the experiments are listed in **Table 1**. Unless stated explicitly, the default parameter values, given in bold are used.

6.2 Experimental Results

Node Reduction Rate. We first study the efficiency of node pruning technique in terms of the node reduction rate. We set threshold ε to the value that optimizes the overall performance (through a tuning processing similar to **Fig. 13**). Specifically, we set ε to 0.01 for the uniform dataset, and ε to 0.005 for the clustered and correlated datasets.

In order to examine the impact of the dimensionality, we used sensor networks with 8000 nodes with dimensionality varying from 2 to 5. **Fig. 8(a)** plots the experiment results for uniform and clustered datasets. We can see that the performance degrades when dimensionality increases. The charts also show that, the NRR in clustered dataset is 25% to 53% higher than uniform dataset. The reason is that, data points in clustered dataset concentrate in several regions, and thus, more sensor nodes are pruned due to mapped data close to query point. To study the influence of the node cardinality, we used a 4-dimensional dataset and varied the network size between 6000 and 9000 nodes. As shown in **Fig. 8(b)**, ECRS in clustered dataset prune about 50% more nodes than uniform dataset. We also see that, the NRR is not much sensitive to the cardinality under both distributions.

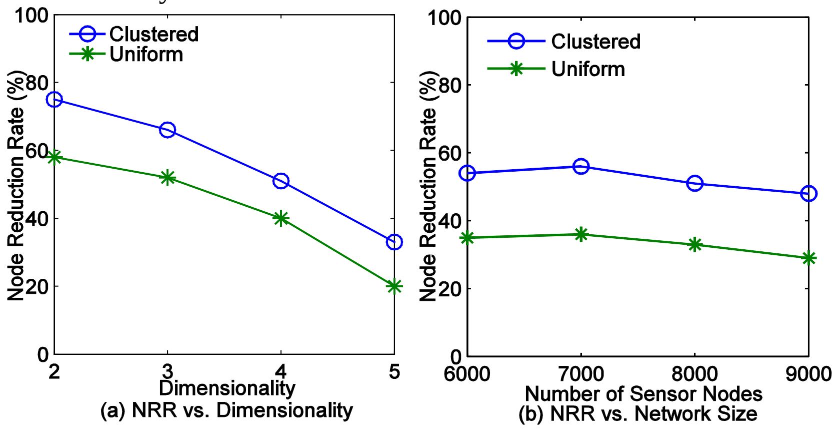


Fig. 8. Data reduction rate (uniform and clustered datasets)

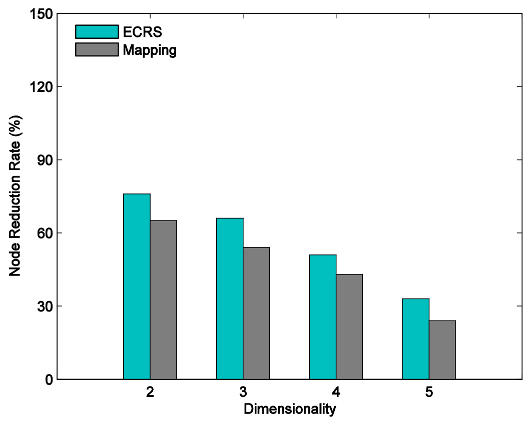


Fig. 9. Data reduction rate of ECRS and Mapping approaches

In **Fig. 9**, we compare ECRS with *Mapping* method for clustered dataset. We can see that ECRS is 15% to 28% more efficient than *Mapping*. The experiments results indicates that our pruning strategy can efficiently reduce the number of nodes that need to report their sensor readings.

Communication Cost. We then proceed to investigate the communication cost. We evaluate the performance of ECRS under three data distributions, i.e., uniform (**Fig. 10**), clustered (**Fig. 11**), and correlated (**Fig. 12**).

Fig.10(a) depicts the communication cost when the dimensionality varies from 2 to 5. We fix the node cardinality to 8000. ECRS transfers more than 45%~130% fewer data than *Basic* due to our node pruning technique employed by ECRS, which greatly reduces the number of nodes that need to report their readings (as shown in **Fig. 8**). We also see that, as the dimensionality increases, the communication cost grows rapidly. There are two reasons. Firstly, an increase of dimensionality leads to the increase of the size of the individual sensor readings as well as the cardinality of the reverse skylines. Secondly, the pruning capability degrades when dimensionality grows.

Fig. 10(b) illustrates the communication cost as a function of the number of sensor nodes for the 4-dimensional dataset. ECRS transfers about 89% fewer data than *Basic* and 23% than *Mapping*. Since a network of a larger size generates more data, the number of reverse skylines as well as the average cost of processing such a query increases. Hence, both approaches transmit more bytes of data when the number of nodes increases.

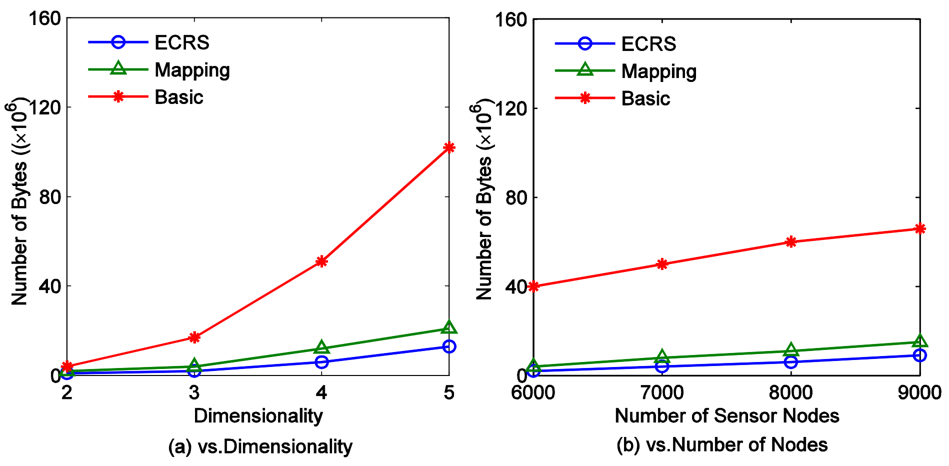


Fig. 10. Communication cost (uniform dataset)

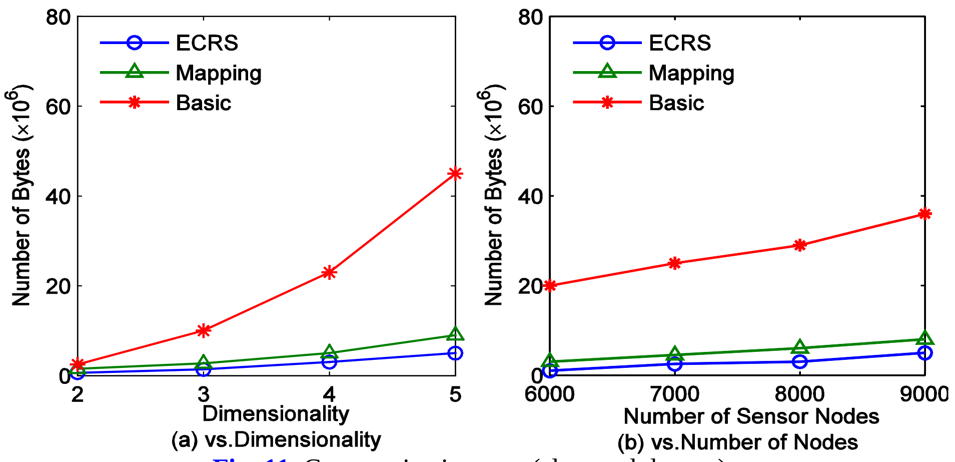


Fig. 11. Communication cost (clustered dataset)

Fig. 11 shows the communication cost of clustered dataset. ECRS shows consistently better performance than *Basic*. For the 3-dimensional dataset, the communication cost is an order of magnitude less than that of *Basic*. We also see that, ECRS outperforms *Basic* more significantly as dimensionality increases. Although communication costs of both algorithm increase with dimenisionality, the cost incresement of *Basic* is much more obvious than that of ECRS.

From Fig. 10 and Fig. 11, we can see that, the communication cost of uniform dataset is higher than that of clustered dataset. The reason is that, more unqualified nodes are pruned for clustered dataset, and therefore fewer sensor readings are transferred. The charts also show that the cost of ECRS is always lower than that of *Mapping*, which proves the effectiveness of our pruning strategy.

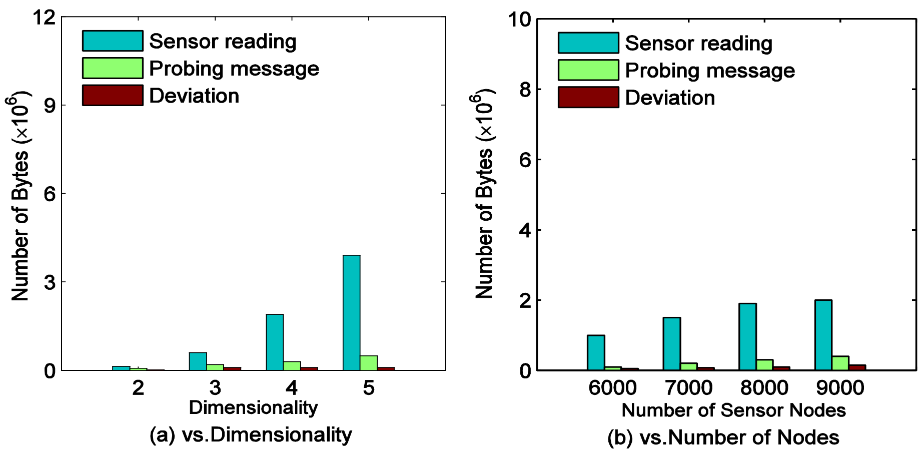


Fig. 12. Communication cost (correlated dataset)

In ECRS, three types of data are transmitted over the network, i.e., sensor readings, deviations, and the probing message. Specifically, both sensor readings and deviations are associated with node ids. The probing message contains the ids of the nodes that need to report their sensor readings. In order to clearly illustrate the effect of transferring these three types of data, we provide a cost break-down analysis. In Fig. 12, we depict the classified communication cost in correlated dataset. The chart shows that cost for transferring sensor readings is the dominator of the overall communication cost (59%~87%). Specifically, as dimensionality increases, the proportion of cost for transferring sensor readings increases (from 59% to 87%, as shown in **Fig. 12(a)**). As the cardinality of the reverse skyline increases with dimensionality while the pruning capability degrades, the cost for transferring multi-dimensional sensor readings grows rapidly with dimensionality. Furthermore, in ECRS, a sensor node needs to report its true deviation only when the one-dimensional deviation is larger than the default value ε. Therefore, the effect of additional cost on the overall communication cost decreases. We only detail the results on correlated dataset here, because ECRS yields similar performances on the other two datasets. From **Fig. 10** to **Fig. 12**, we can see that, although additional communication cost is needed for collecting deviations and sending the probing message in ECRS, the overall cost decreases as a result of the significant reduction of the number of transferred sensor readings.

Effects of the parameter. In the last experiment, we evaluate the performance of ECRS under different values of threshold ε. In ECRS, ε is set as the default value of deviation, and a sensor node reports its deviation as mapping information only when its deviation is larger than ε. **Fig. 13(a)** evaluate the communication cost of ECRS in uniformly distributed dataset when we vary ε from 0.006 to 0.04. We used a 4-dimensional dataset with 8000 points. When ε is relatively small, e.g., $\varepsilon< 0.01$, the performance degrades if ε decreases. A smaller ε does not necessarily ensure a better performance. The reason is that when ε goes smaller, unqualified nodes need to report their true deviations for mapping information collection. While $\varepsilon>0.02$, the communication cost increases when ε goes larger. This is because, a larger ε leads sensor nodes to be represented with ε-hypersquares with greater volumes, and thus, more unqualified nodes are not pruned based on their mapping information. **Fig. 13(b)** shows the effect of ε on the communication cost when data distribution is clustered. When ε is close to 0.005, the communication cost is under 5 million and ECRS has relatively good performance. The correlated distribution exhibit similar behavior, and hence, are omitted here.

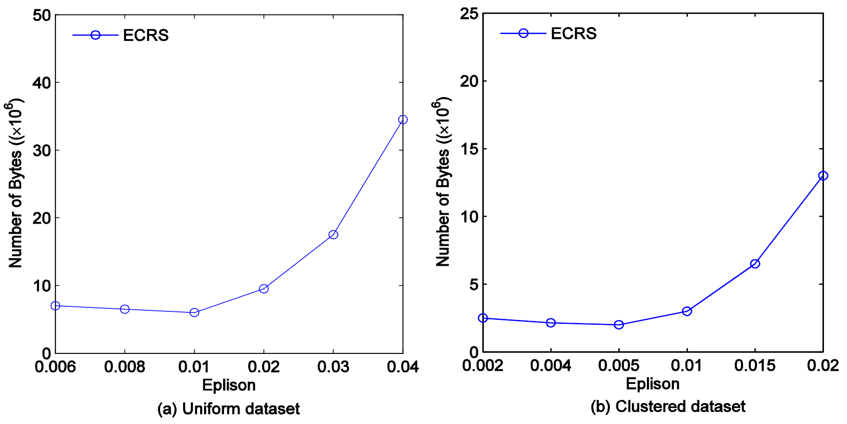


Fig. 13. Quality vs. threshold

Summary. From the experiments, we can conclude that for continuous reverse skyline, our proposed ECRS approach can efficiently reduce the number of nodes that need to send their sensor readings. ECRS consistently outperforms the the coutinuous version of algorithm of [21] in terms of communication cost under various network configurations, and the superiority is more significant as the dimension increases.

7. Conclusion

In this paper, we have proposed ECRS, an energy-efficient approach that processed the continuous reverse skyline queries in wireless sensor networks. The basic idea is to suppress the updates from nodes not contributing to the reverse skyline result. A data mapping scheme is used to estimate current sensor readings, and identify nodes that produced reverse skylines. Node pruning techniques are proposed to remove unqualified nodes while guaranteeing the correctness of the answer. Our simulation revealed the effectiveness and superior efficiency of the ECRS. Some extensions can be considered. An interesting topic is to examine the effectiveness of node pruning technique in other query operators, such as *k* nearest neighbor query and range query. Furthermore, it is equally exciting to study secure reverse skyline query and the reverse skyline queries in various subspaces.

References

[1] A.Silberstein, R.Braynard, C.Ellis, K.Munagala, and J.Yang, “A sampling-based approach to optimizing top-k queries in sensor networks,” in *Proc. of 22nd International Conference on Data Engineering*, pp. 68-68, April 3-8, 2006. Article (CrossRef Link)

[2] Y.Yao, X.Tang, and E.Lim, “Localized monitoring of kNN queries in wireless sensor networks,” *The VLDB Journal*, vol. 18, no. 1, pp.99-117, January, 2009. Article (CrossRef Link)

[3] Y.Zhang, X.Sun, and B. Wang, “Efficient algorithm for k-barrier coverage based on integer linear programming,” *China Communications*, vol.13, no.7, pp.16-23, September, 2016. Article (CrossRef Link)

[4] S.Börzsönyi, D.Kossmann, and K.Stocker, “The skyline operator,” in *Proc. of 17th International Conference on Data Engineering*, pp.421-430, April 2-6, 2001. (CrossRef Link)

[5] D.Papadias, Y.Tao, G.Fu, and B.Seeger, “An optimal and progressive algorithm for skyline queries,” in *Proc. of ACM SIGMOD International Conference on Management of Data*, pp.467-478, June 9-12, 2003. Article (CrossRef Link)

[6] E.Dellis and B.Seeger, “Efficient computation of reverse skyline queries,” in *Proc. of 33rd International Conference on Very Large Data Bases*, pp.291-302, September 23-27, 2007. Article (CrossRef Link)

[7] Y.Gao, Q.Liu, B.Zheng, and G.Chen, “On efficient reverse skyline query processing,” *Expert Systems with Applications*, vol.41, pp.3237-3249, June, 2014. Article (CrossRef Link)

[8] M.S.Islam, R. Zhou, and C. Liu, “On answering why-not questions in reverse skyline queries, ” in *Proc. of 29nd International Conference on Data Engineering*, pp.973-984, April 8-11, 2013. Article (CrossRef Link)

[9] X.Wu, Y.Tao, R.C.-W.Wong, L.Ding, and J.X.Yu, “Finding the influence set through skylines,” in *Proc. of 12th International Conference on Extending Database Technology: Advances in Databse Technology*, pp.1030-1041, March 23-26, 2009. Article (CrossRef Link)

[10] A.Arvanitis, A.Deligiannakis, and Y.Vassiliou, “Efficient influence-based processing of market research queries,” in *Proc. of 21st ACM international conference on Information and knowledge management*, pp.1193-1202, October 29 to November 2, 2012. Article (CrossRef Link)

[11] X.Lian and L.Chen, “Monochromatic and bichromatic reverse skyline search over uncertain databases,” in *Proc. of ACM SIGMOD International Conference on Management of Data*, pp.213-226, June 10-12, 2008. Article (CrossRef Link)

[12] T.Jiang, Y.Gao, B.Zhang, D.Lin, and Q.Li, “Monochromatic and bichromatic mutual skyline queries,” *Expert systems with applications*, vol.41, pp.1885-1900, March, 2014. Article (CrossRef Link)

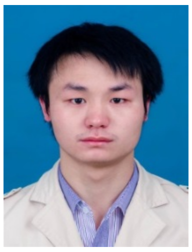
[13] Y.Gao, Q.Liu, L.Mou, G.Chen, and Q.Li, “On processing reverse k-skyband and ranked reverse skyline queries,” *Information science*, vol.293, pp.11-34, February, 2015. Article (CrossRef Link)

[14] Z.Wang, Y.Gao, Q.Liu, X.Miao, Q.Li, and C.Li, “Efficient group-by reverse skyline computation,” *World Wide Web*, pp.1-27, November, 2015. Article (CrossRef Link)

[15] J.Xin, Z.Wang, M.Bai, and G.Wang, "Reverse Skyline Computation over Sliding Windows," *Mathematical Problems in Engineering*, vol.2015, 2015. Article (CrossRef Link)
[16] J.-K.Min, "Efficient reverse skyline processing over sliding windows in wireless sensor networks," *International journal of distributed sensor networks*, Vol.2015, January, 2015. Article (CrossRef Link)
[17] P.M.Deshpande, and P.Deepak, "Efficient Reverse skyline retrieval with arbitrary non-metric similarity measures," in *Proc. of ACM SIGMOD International Conference on Management of Data*, pp.319-330, June 12-16, 2011. Article (CrossRef Link)
[18] Y.Park, J.-K. Min, and K. Shim, "Parallel computation of skyline and reverse skyline queries using mapreduce," in *Proc. of VLDB Endowment*, vol.6, no.14, pp.2002-2013, 2013. Article (CrossRef Link)
[19] M.Bai, J.Xin, and G.Wang, "Probabilistic reverse skyline query processing over uncertain data stream," *Database Systems for Advanced Applications*, vol.7239, pp.17-32, 2012. Article (CrossRef Link)
[20] J.Lim, K.Bok, Y.Lee, and J.Yoo, "A continuous reverse skyline query processing for moving objects," in *Proc. of International Conference on Big Data and Smart Computing*, pp.66-71, January 15-17, 2014. Article (CrossRef Link)
[21] G.Wang, J.Xin, L.Chen, and Y.Liu, "Energy-efficient reverse skyline query processing over wireless sensor networks," *IEEE Transactions on Knowledge and Data Engineering*, vol.24, no.7, pp.1259-1275, March, 2012. Article (CrossRef Link)
[22] H.Chen, S.Zhou, and J.Guan, "Towards energy-efficient skyline monitoring in wireless sensor networks," in *Proc. of* 4th *European Conference on Wireless Sensor Networks*, pp.101-116, January 29-31, 2007. Article (CrossRef Link)
[23] Y.Kwon, J.-H.Choi, Y.D.Chung, and S.Lee, "In-network processing for skyline queries in sensor networks," *IEICE Transactions on Communications*, Vol.E90-B, No.12, pp.3452-3459, 2007. Article (CrossRef Link)
[24] J.Xin, G.Wang, L.Chen, X. Zhang, and Z.Wang, "Continuously maintaining sliding window skylines in a sensor network," in *Proc. of 12th International Conference on Database Systems for Advanced Applications*, pp.509-521, April 9-12, 2007. Article (CrossRef Link)
[25] W.Liang, B.Chen, and J.X.Yu, "Energy-Efficient Skyline Query Processing and Maintenance in Sensor Networks," in *Proc. of 17th ACM Conference on Information and Knowledge Management*, pp.1471-1472, October 26-30, 2008. Article (CrossRef Link)
[26] I.-F.Su, Y.-C.Chung, C.Lee, and Y.-Y.Lin, "Efficient Skyline Query Processing in Wireless Sensor Networks," *Journal of Parallel and Distributed Computing*, Vol.70, No.6, pp.680-698, June, 2010. Article (CrossRef Link)
[27] O.Younis, and S.Fahmy, "HEED: a Hybrid, energy-Efficient, distributed clustering approach for ad hoc sensor networks," *IEEE Transactions on mobile computing*, vol.3, no.4, pp.366-379, October, 2004. Article (CrossRef Link)
[28] A.Vlachou, C.Doulkeridis, Y.Kotidis, and M.Vazirgiannis, "SKYPEER: Efficient subspace skyline computation over distributed data," in *Proc. of 23rd International Conference on Data Engineering*, pp. 416-425, April 15-20, 2007. Article (CrossRef Link)

Bo Yin received her B.S. degree in communication engineering, M.S. degree in communication and information system, and Ph.D.degree in computer application technology from Hunan University, Changsha, China, in 2005, 2008, and 2013 respectively. She is currently a lecturer at School of Computer and Communication Engineering, Changsha University of Science and Technology, Changsha, China. Her current research interests include data management and mining, parallel and distributed computing, and wireless communication systems.

Siwang Zhou received his M.S. degree in computer application technology and Ph.D.degree in control science and engineering from Hunan University, Changsha, China, in 2009, and 2015 respectively. He is currently an associate professor at College of Information Science and Engineering, Hunan University, Changsha, China. His research interests include machine learning, database system, and wireless communication systems.



Shiwen Zhang received his Ph.D. degree in computer application technology from Hunan University, Changsha, China, in 2015. He is currently a lecturer at Hunan University of Science and Technology, Xiangtan, China. His research interests include security and privacy issues in social networks, cloud computing, sensor networks, and data mining.

Ke Gu received his Ph.D. degree in School of Information Science and Engineering from Central South University in 2012. He is currently a Lecturer at Changsha University of Science and Technology, Changsha, China. His research interests include cryptography, network and information security.

Fei Yu received his Ph.D. degree in computer science and technology from Hunan University, Changsha, China, in 2013. He is currently a lecturer at Changsha University of Science and Technology, Changsha, China. His research interests include wireless communication systems and radio frequency integrated circuits design.