Progressive Skyline Query Evaluation and Maintenance in Wireless Sensor Networks

Baichen Chen  
Australian National University  
Canberra, ACT 0200, Australia  
baichen@cs.anu.edu.au

Weifa Liang  
Australian National University  
Canberra, ACT 0200, Australia  
wliang@cs.anu.edu.au

Jeffrey Xu Yu  
Chinese Univ. of Hong Kong  
Shatin, NT, Hong Kong  
yu@se.cuhk.edu.hk

ABSTRACT
Skyline query has been received much attention due to its wide application backgrounds for multi-preference and decision making. In this paper we consider skyline query evaluation and maintenance in wireless sensor networks. We devise an evaluation algorithm for finding skyline points progressively and a maintenance algorithm for skyline maintenance incrementally. We also conduct extensive experiments by simulations to evaluate the performance of the proposed algorithms on various datasets. The experimental results show that the proposed algorithms significantly outperform existing algorithms in terms of network lifetime prolongation.

Categories and Subject Descriptors
H.2.4 [Database Management]: Query Processing

General Terms
Algorithm, Experimentation, Performance

Keywords: WSN, skyline query, query optimization

1. INTRODUCTION
To support data query processing in wireless sensor networks (WSNs), several DB systems like TinyDB [5] and Cougar [11] have been developed. However, they only support some basic operators. With further development of hardware techniques in sensors and wide applications of WSNs, it is becoming urgent that WSNs can also support more complicated queries like skyline queries. In this paper we focus on skyline query evaluation and maintenance in WSNs.

Extensive studies on skyline query in centralized databases [3, 4, 7] and distributed databases [1, 9] have been conducted in recent years, existing algorithms however are not applicable to WSNs due to the unique characteristics imposed on sensor networks including limited storage, energy-limited battery, slow processing capability, narrow communication bandwidth, etc. Several studies of skyline query in WSNs have been conducted in the past [2, 10], they mainly focused on optimizing the total energy consumption alone rather than the tradeoff between the total energy consumption and the maximum energy consumption among the sensors, while the maximum energy consumption among the sensors usually is the bottleneck of the network lifetime [6]. Therefore, energy-efficient skyline query evaluation and maintenance in WSNs poses great challenges.

2. PRELIMINARIES
We consider a wireless sensor network consisting of \( n \) stationary sensors randomly deployed in a region of interest, and each sensor is equipped with \( d \) sensing devices to measure \( d \) attribute values. Each data point consists of \( d \) attribute values, referred to as a \( d \)-dimensional data point. There is a base station with unlimited energy supply serving as the gateway between the sensor network and users. Each sensor can communicate with the other sensors and the base station via one or multi-hop relays. To transmit a message containing \( k \) bytes of data from one sensor to another, the amounts of transmission energy consumed at the sender are \( \rho_t + R \times k \), and the amounts of reception energy consumed at the receiver are \( \rho_r + r_e \times k \), where \( \rho_t \) and \( \rho_r \) are the sums of energy overhead on handshaking and transmitting and receiving header part of the message, \( R \) and \( r_e \) are the amounts of transmission and reception energy per byte, respectively. Each data point is represented by \( 4 \times d \) bytes in this paper. The energy-efficient skyline query evaluation in sensor networks can be implemented through in-network processing paradigm by building a routing tree \( T \) rooted at the base station \( r \) and spanning all sensors.

Given a wireless sensor network \( G(V, E) \) with base station \( r \), where \( V \) is the set of sensors and \( E \) is the set of links. Assume that each sensor \( v \) in \( V \) has a set of snapshot points \( P(v) \) generated during a given time interval, and \( P = \cup_{v \in V} P(v) \) forms the entire dataset. The skyline query on \( P \) is to find a subset of \( P \), referred to as \( SK(P) \), in which the points cannot be dominated by any other points in \( P \). Skyline maintenance is to maintain the skyline dynamically within sliding window environments. The data generated by sensors are treated as the boundless streaming data, it thus is impossible to perform a skyline query on all generated points so far due to the huge volume of data involved. Instead, a sliding window skyline will only consider the skyline on the set of points generated within the current window. Let \( W \) be the length of sliding window which is equal to the lifespan of each point. The skyline maintenance at each time step \( t \) is to evaluate the skyline on \( P_t = \{ p \mid t - p.time \leq W \} \), where \( p.time \) is the generated time step of point \( p \).
For skyline evaluation, we introduce a simple algorithm, the **skyline merge** algorithm. If sensor \( v \) is a leaf node, it sends the skyline of local points to its parent; Otherwise, sensor \( v \) computes the skyline on the set of points at \( v \) including the points generated at \( v \) and forwarded by its children, and transmits the new skyline to its parent. Finally, the base station \( r \) obtains the skyline on the set of received points, which is the skyline on the set \( P \).

The following observation is the cornerstone of the rest of the paper, which will be used later.

**Definition 1.** \([3, 7] \) Suppose \( p = (p_1, p_2, ..., p_d) \) is a \( d \)-dimensional point, the radius of \( p \) \( R(p) \) is defined as the Euclidean distance between \( p \) and the origin \( o = (0, 0, \ldots, 0) \), i.e., \( R(p) = \sqrt{\sum_{i=1}^{d} p_i^2} \).

**Observation 1.** \([7] \) Let \( R(p) \) and \( R(q) \) be the radii of points \( p \) and \( q \). If \( R(p) \leq R(q) \), \( p \) cannot be dominated by \( q \).

### 3. Dynamic Radius-Partition Based Algorithm

In this section an evaluation algorithm **DRP** (Dynamic Radius Partition) is devised, which first partitions the dataset \( P \) into disjoint subsets based on dynamic partition radius, followed by evaluating the skyline on each subset progressively through the use of found skyline points as the global filter to filter out unlikely skyline points in the rest of subsets from transmission within the network.

#### 3.1 Dynamical Dataset Partition

Intuitively, algorithm DRP proceeds in a number of iterations \( k \), and \( k \) is dynamically determined by the data distribution in the dataset. The dataset \( P \) is partitioned into several disjoint subsets and the partition radius in each iteration is determined as follows.

Denote by \( LSK(v)_i \), the skyline of the points at sensor \( v \) and the received points from the children of \( v \) in the \( i \)th iteration, \( 1 \leq i \leq k \). Let \( LF(v)_i \), be the **Local Filter** at sensor \( v \) and \( UR(v)_i \), the **upper bound of the transmission radius** of sensor \( v \) in the \( i \)th iteration. If sensor \( v \) is a leaf node, \( UR(v)_i \) is the maximum radius of all points forwarded by \( v \). Otherwise, \( UR(v)_i \) is calculated as follows. Assume that sensor \( v \) has \( d_v \) children, \( u_1, u_2, \ldots, u_{d_v} \). In the \( i \)th iteration, sensor \( v \) receives the points from each child \( u_j \) and calculates \( LSK(v)_i \), \( 1 \leq j \leq d_v \). Then, \( UR(v)_i \) is the minimum among the \( d_v \) maximum radii of the points sent by the children of \( v \) and the maximum radii of the points in \( LSK(v)_i \). Obviously, \( UR(v)_i \leq UR(u)_i \) when \( u \) is a descendant of \( v \). Thus, \( UR(r)_i \), \( UR(v)_i \) for any sensor \( v \in V \). \( UR(r)_i \) is the **partition radius** of the \( i \)th partitioned subset \( P_i \), i.e., the radii of all the points in \( P_i \) are no greater than \( UR(r)_i \). Therefore, the dataset is dynamically partitioned by \( UR(r)_i \), \( 1 \leq i \leq k \).

Assuming that algorithm DRP has performed the first \((i-1)\)th iterations already, and now proceeds the \( i \)th iteration as follows.

Each sensor \( v \) filters out the local points dominated by any point in \( LF(v)_i \). If sensor \( v \) is a leaf node, it transmits \( LSK(v)_i \), to its parent, where \( LSK(v)_i \) is the skyline of the points at sensor \( v \). Otherwise, it calculates \( LSK(v)_i \), and the upper bound of the transmission radius of sensor \( v \), \( UR(v)_i \). Sensor \( v \) then transmits the points in \( LSK(v)_i \), whose radii are no greater than \( UR(v)_i \), to its parent. Having received the points from all children, the base station \( r \) calculates \( LSK(r)_i \) and \( UR(r)_i \). The newly found skyline in the \( i \)th iteration is \( SK_i = \{ p \mid p \in LSK(r)_i, R(p) \leq UR(r)_i, \forall q \in \bigcup_{j=1}^{i-1} SK_j, q \neq p \} \), where \( q \neq p \) means that point \( q \) cannot dominate point \( p \). Some points in \( SK_i \) will be chosen for broadcast as the global filter to update the local filter of each sensor \( v \). In case \( SK_i \) is empty, the **skyline merge** algorithm will be applied to find the remaining skyline points. The algorithm terminates and the number of iterations \( k = i + 1 \) is determined. The skyline on \( P \) is \( SK(P) = \bigcup_{i=1}^{k} SK_i \). In the following we detail which points in \( SK_i \) will be chosen.

#### 3.2 Global Filter Broadcasting

Having obtained \( SK_i \) at \( r \), a simple way to update the local filter of each sensor \( v \) is to broadcast all the points in \( SK_i \) to it. However, this naive approach will incur a much more energy overhead than needed. On the other hand, if none of the newly found global skyline points is broadcast, the local filter of each sensor cannot filter out as many unlikely skyline points as possible without the help by the newly skyline points. To this end, we propose a method to tradeoff the energy consumption between broadcasting and data transmission, based on the volume of the **efficient dominance region** of points. The efficient dominance region of a point \( p \) is the subspace of the dominance region of \( p \) that has not been covered by the dominance regions of previously found skyline points and all the points inside are not yet examined.

Denote by \( EDR(p) \), an approximate volume of the efficient dominance region of point \( p \) at the \( j \)th dimension, \( 1 \leq j \leq d \). Let \( GSF_j \) be the set of chosen skyline points to broadcast after the \( j \)th iteration. Given a set \( P \) of \( d \) dimensional points, let \( Max = (max_1, \ldots, max_d) \) and \( MinSK(i) = (minSK(i)_1, \ldots, minSK(i)_d) \) be the virtual points, where \( max_j \) is the maximum value at the \( j \)th dimension of the points in \( P \), and \( minSK(i)_j \) is the minimum value at the \( j \)th dimension of all found skyline points in previous iterations. The approximate volume of the efficient dominance region of point \( p = (p_1, p_2, ..., p_d) \) in \( SK(P) \) at the \( j \)th dimension is \( EDR(p)_j = minSK(i)_j - p_j \ast (\prod_{k=1, k \neq j} (max_k - p_k) - \prod_{k=1, k \neq j} UR(r)_k) \). For each dimension \( j \), the point \( p \) in \( SK_i \) with the maximum value of \( EDR(p)_j \) is chosen and added to \( GSF_j \), if \( p \notin GSF_j \) and \( EDR(p)_j > 0 \). As a result, at most \( d \) skyline points (at most one point chosen in each dimension) are chosen for broadcast to update the local filters.

### 4. Skyline Maintenance Algorithm

Once the initial skyline is found, what followed is to monitor the skyline with time progress. In this section we propose a novel algorithm **MSM** to maintain the skyline under a sliding window environment incrementally.

#### 4.1 Overview of the Maintenance Algorithm

Let \( t_0 \) be the time step at which the update starts and \( P_0 \) is the initial set of points. The initial skyline, \( SK(P_0) \), is obtained by the proposed algorithm **DRP**. Algorithm **MSM** is used to maintain the skyline in the WSN at each time step \( t \) with \( t > t_0 \), which consists of two phases. One is to update the new skyline points by traversing the sensors in
the routing tree in the bottom-up fashion. Another is that, the base station \( r \) determines which found skyline points so far to be broadcast to update the local filter of each sensor.

In the following, we detail these two phases.

### 4.2 Updating New Skyline Points

At time step \( t \), each sensor first updates set \( P(v) \), by removing expired points and adding newly generated points, where \( P(v) \) is the set of points at sensor \( v \). Each sensor also removes expired points from its local filter \( LF(v) \). If point \( p \) in \( P(v) \) is dominated by another point \( q \) in either \( P(v) \) or \( LF(v) \), and \( q.t \geq p.t \), \( p \) is safely removed. Let \( CSK(v) \) be the set of remaining points in \( P(v) \) that are not dominated by any point in \( LF(v) \). The skyline merge algorithm is then applied on set \( \cup_{v \in V}CSK(v) \) to obtain the skyline \( NewSK_t \), and the skyline on the union of \( NewSK_t \) and the subset of \( SK(P_{t-1}) \) of non-expired points is the skyline of the dataset at time step \( t \), i.e., \( SK(P_t) \). The base station \( r \) then determines whether to broadcast some skyline points obtained since last broadcast to update the local filters, and which skyline points should be broadcast. In the following, this issue will be addressed.

### 4.3 Broadcasting

We first address which found skyline points to be chosen for broadcast to avoid excessive energy consumption by broadcasting all the skyline points. To do so, two factors are to be taken into account: one is that a point with longer remaining lifespan will potentially filter out much more points; another is that a point with larger volume of efficient dominance region will filter out more points. Specifically, denote by \( GSF_t \) the set of points broadcast at time step \( t' \) and \( GSF_t(t) = \{ p | p \in GSF_t, p.t > t - W \} \), where \( t \) is the current time step. Let \( \text{Min}(t) = (\min SK(t_1), \ldots, \min SK(t_d)) \) be a \( d \)-dimensional virtual point, where \( \min SK(t_i) \) is the minimum value at the \( i \)th dimension among all the points in \( GSF_t(t) \) at time step \( t \). Let \( \text{margin}(p,t_i) \) represent the distance from point \( p \) to the region being dominated by the skyline points in \( GSF_t(t) \) at the \( i \)th dimension. If \( \min SK(t_i) > p_i \), \( \text{margin}(p,t_i) = \min SK(t_i) - p_i \); otherwise, \( \text{margin}(p,t_i) = \infty \). The approximate evaluation function of the efficient dominance region \( EDR(p,t_i) = \text{margin}(p,t_i) \times \prod_{i=1,i\neq i}(p_i) \). A point \( p \) with smaller \( EDR(p,t_i) \), is able to filter out more points. For every point \( p \) in set \( SK(P_t) - GSF_t(t) \), define the weight of \( p \) as \( w(p) = EDR(p,t_i)/(W - t + p.time) \), at each dimension \( i \) is calculated, and the point \( p \) is added to \( GSF_t \) if \( w(p) = \min(w(q)), q \in SK(P_t) - GSF_t(t) \) and \( p \notin GSF_t \), where \( W - t + p.time \) is the remaining lifespan of \( p \).

We then investigate when the base station \( r \) broadcasts \( GSF_t \) into the sensor network to update local filters. Assume that \( r \) broadcasts \( GSF_t \) into the sensor network. At each time step \( t \), the base station \( r \) computes \( GSF_t \) as mentioned above. Triggering a broadcast depends on whether the gain of filtering out more points by the updated local filters outweighs the broadcast overhead. We now analyze the gain of broadcasting \( GSF_t \). Assume that \( S_t \) is the set of points received by the base station \( r \) and dominated by the points in \( GSF_{t-1} \). If \( GSF_{t-1} \) was broadcast at time step \( t - 1 \), all the points in \( S_t \) will be filtered out from the transmission and thus the total amount of saved energy is at least \( \sum_{p \in GSF_t} h(p,v)4d(R + r_e) \), where \( h(p,v) \) is the number of hops between \( r \) and sensor \( v \) in which point \( p \) is originally located, which will be used to predict that it will bring the similar amounts of energy saving in future if \( GSF_t \) is broadcast, and \( GSF_t \) is expected to be part of the local filter within the time interval \( (t, t + T_t) \), where \( T_t \) is the average remaining lifespan of the points in \( GSF_t \). Thus, the total amount of energy saving \( E_{save}(t) \) during \( (t, t + T_t) \) is as follows,

\[
E_{save}(t) \geq \sum_{p \in S_t} h(p,v)4d(R + r_e)T_t \tag{1}
\]

We finally evaluate the overhead on broadcasting \( GSF_t \) at time step \( t \), which includes the energy consumption overhead on broadcasting \( GSF_t \) into the sensor network and the unpaid energy saving overhead on the last broadcast at time step \( t' \) with \( t' < t \). The total amount of energy overhead on broadcasting to the whole sensor network is no greater than \( (d + 4d \ast (R + r_e) + (\rho_1 + \rho_2)) \ast n \), where \( n \) is the number of sensors, because every sensor will receive a message from its parent and broadcast the message to its children. On the other hand, \( GSF_t \) is expected to be part of the local filter of each sensor from \( t' \) to \( t' + T_{t'} \), where \( T_{t'} \) is the average remaining lifespan of the points in \( GSF_{t'} \). However, if \( r \) broadcasts \( GSF_t \) at time step \( t \) during \( (t', t' + T_{t'}) \), this means that the last broadcast does not deliver its promised energy saving from \( t + 1 \) to \( t' + T_{t'} \). Thus, the amounts of unpaid energy are \( \sum_{p \in GSF_{t'}} h(p,v)4d(R + r_e) \), \( t' + T_{t'} - t \), where \( S_{t'} \) is the set of points being forwarded to the base station \( r \) and dominated by \( GSF(t') \) at time step \( t' \). Accordingly, the overhead on broadcasting \( GSF_t \), \( \text{Cost}(t) \), is thus

\[
\text{Cost}(t) \leq (4d^2(R + r_e) + (\rho_1 + \rho_2))n + \sum_{p \in GSF_{t'}} h(p,v)4d(R + r_e)(T_{t'} + t' - t) \tag{2}
\]

If \( t < T_{t'} + t' \), combined inequalities (1) and (2), we have

\[
\sum_{p \in GSF_{t'}} h(p,v)4d(R + r_e)(T_{t'} + t' - t) \geq (4d^2(R + r_e) + (\rho_1 + \rho_2))n + \sum_{p \in S_{t'}} h(p,v)4d(R + r_e)(T_{t'} + t' - t) \tag{3}
\]

Otherwise,

\[
\sum_{p \in GSF_{t'}} h(p,v)4d(R + r_e)(T_{t'} + t' - t) \geq (4d^2(R + r_e) + (\rho_1 + \rho_2))n \tag{4}
\]

Thus, the base station \( r \) will trigger a broadcast if the inequality either (3) or (4) is met, depending on whether \( t < T_{t'} + t' \).

### 5. PERFORMANCE EVALUATION

In this section we evaluate the performance of algorithms DRP and MSM against existing algorithms in terms of the total energy consumption and the maximum energy consumption among sensors. We assume that the sensor network containing 500 sensors with each being 10 meters transmission range is used to monitor a 100m \times 100m region of interest, and the base station is located at the square center. The energy overheads on transmitting and receiving a header and handshake are \( \rho_1 = 0.4608 \text{ mJ} \) and \( \rho_2 = 0.1152 \text{ mJ} \). The energy consumptions of transmitting and receiving one byte are \( R = 0.0144 \text{ mJ} \) and \( r_e = 0.00576 \text{ mJ} \), respectively. In our experiments, the datasets are the real sensing datasets obtained by the Intel Lab at UC Berkeley [12].

We first evaluate the performance of algorithm DRP against existing algorithms by varying dimensionality \( d \) from 2 to 4. We refer to the dynamic filter algorithm by Huang et al. [2]
as algorithm DF, the single point filter algorithm and grid index filter algorithm by Xin et al. [10] as algorithms TF and GI, respectively. Fig. 1(a)-(d) show the total energy consumption and the maximum energy consumption among the sensors by various algorithms on real datasets. It can be seen that the proposed algorithm DRP outperforms the others significantly.

![Figure 1: The performance of evaluation algorithms](image)

We then study the performance of different algorithms for skyline maintenance on real sensing datasets. The window length is set \( W = 300 \). Fig. 2 plots the performance of different maintenance algorithms within sliding windows from time step 301 to 3,600. From Fig. 2(a)-(f), algorithm MSM has the best performance among all the algorithms in terms of the total energy consumption and the maximum energy consumption among the sensors. With time going by, the performance gap between algorithm MSM and algorithm GI becomes bigger and bigger. Thus, algorithm MSM delivers a longer network lifetime than that of existing algorithms.

![Figure 2: The performance of maintenance algorithms](image)

6. CONCLUSIONS

In this paper we first devised an algorithm DRP for skyline query evaluation that returns the skyline points progressively. We then proposed an energy-efficient incremental algorithm MSM for skyline maintenance within sliding window environments. We also conducted experimental simulations on real sensing datasets. The experimental results show that the proposed algorithms significantly outperform existing algorithms in terms of various performance metrics.

7. REFERENCES


