Wireless Netw (2012) 18:985–1004
DOI 10.1007/s11276-012-0446-z

Energy-efficient skyline query optimization in wireless sensor networks

Baichen Chen · Weifa Liang · Jeffrey Xu Yu

Published online: 12 May 2012

Abstract With the deployment of wireless sensor networks (WSNs) for environmental monitoring and event surveillance, WSNs can be treated as virtual databases to respond to user queries. It thus becomes more urgent that such databases are able to support complicated queries like skyline queries. Skyline query which is one of popular queries for multi-criteria decision making has received much attention in the past several years. In this paper we study skyline query optimization and maintenance in WSNs. Specifically, we first consider skyline query evaluation on a snapshot dataset, by devising two algorithms for finding skyline points progressively without examining the entire dataset. Two key strategies are adopted: One is to partition the dataset into several disjoint subsets and produce the skyline points in each subset progressively. Another is to employ a global filter that consists of some skyline points in the processed subsets to filter out unlikely skyline points from the rest of unexamined subsets. We then consider the query maintenance issue by proposing an algorithm for incremental maintenance of the skyline in a streaming dataset. A novel maintenance mechanism is proposed, which is able to identify which skyline points from past skylines to be the global filter and determine when the global filter is broadcast. We finally conduct extensive experiments by simulations to evaluate the performance of the proposed algorithms on both synthetic and real sensing datasets, and the experimental results demonstrate that the proposed algorithms significantly outperform existing algorithms in terms of network lifetime prolongation.

Keywords Wireless sensor network · Progressive algorithms · Skyline query · Energy conservation

B. Chen · W. Liang (✉)
Research School of Computer Science, The Australian National University, Canberra ACT 0200, Australia
e-mail: wliang@cs.anu.edu.au

J. X. Yu
Department of System Engineering and Engineering Management, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong

1 Introduction

To support data query processing in wireless sensor networks (WSNs), several DB systems based on WSNs including TinyDB [21] and DB Cougar [32] have been developed in past several years. These DB systems enable supporting some basic database operators such as `SUM`, `MIN`, `AVG`, and so on, due to the miniature size of sensor nodes and unique constraints imposed on sensors including limited storage, powered by energy-limited batteries, slow processing capabilities, small communication bandwidths. With the further development of hardware in sensors and WSN applications, it is becoming urgent that WSNs are able to support more complicated queries like self-join [31], top-k [29], and skylines [7, 8, 17, 30]. In this paper we focus on the skyline query, which has been received much attention recently by the database community due to its wide application for multi-criteria decision making.

Skyline query in WSNs can be used to monitor the extreme sensed data under multiple criteria. For example, scientists can deploy a WSN to monitor air pollution of a region of interest, where the sensors sense the concentration of poisonous gases like CO and SO_2. For example, the places with high concentration of CO or SO_2 are seriously polluted, while the places with high concentration of both CO and SO_2 can also be regarded to suffer serious pollution. A skyline

query issued to this WSN can identify such places for environmental monitoring purpose. Another application example of skyline query is for monitoring bushfires, where each sensor in the WSN can sense the temperature, humidity and smoking density about its vicinity. In a bushfire, the fieriest fire will cause the vicinity of the sensor with high temperature, low humidity, and high smoking density. The places with low temperatures but high smoking densities are also considered to be dangerous. To identify these dangerous places, the fire fighters can issue a skyline query to the WSN and the system will respond to the request by returning the skyline points as the result.

Generally, the skyline query can be defined as follows. Assume that $p = (p_1, \ldots, p_d)$ and $q = (q_1, \ldots, q_d)$ are two d-dimensional *points*, where p_i and q_i are the sensed values of ith dimension of p and q. Each point corresponds to a sensor and each of its different sensing device readings corresponds to a dimension reading. A point p is *dominated* by another point q, denoted by $q \prec p$, if q is no worse than p on all d dimensions and q is *strictly better* than p on at least one dimension, i.e., $\forall i \in \{1, \ldots, d\}, q_i \leq p_i$ and $\exists j \in \{1, 2, \ldots, d\}, q_j < p_j$. Without loss of generality, in this paper we say that "the better" means "the smaller". Given a set S of points, a point p in S is a *skyline point* of S if p is not dominated by any other points in S. The *skyline query* on S retrieves all skyline points in it.

1.1 Related work

Most previous studies on skyline query focused on centralized databases by assuming that the data is stored in a centralized database [3, 9, 13, 15, 22, 25]. The other work dealt with skyline queries under various computational environments, including skyline processing over data streaming [18], top-k skyline with the maximum number of dominated points [19], spatial skyline [16], skyline with partially ordered domains [4], and probabilistic skyline on a set of uncertain data points [23]. Beyond the centralized databases, skyline query has also been exploited in decentralized databases such as the World Wide-Web [2], `CAN` P2P networks [28], `BATON` P2P networks [6] and P2P systems with different topological structures [27].

Although extensive studies on skyline query in traditional databases have been conducted, the existing algorithms are not applicable to WSNs due to unique constraints imposed on WSNs as follows. First, the centralized data structures like the R-tree employed in centralized databases for skyline query processing no longer exist in the WSN environment so that the algorithms based on these centralized data structures are not applicable to WSNs. Second, to prolong the network lifetime, the energy consumption rather than the query response time or space in traditional databases used is the main optimization objective in WSNs, because the battery-powered sensors will quickly become inoperative due to large quantity of energy consumption, if all data are sent to the base station through multi-hop relays, and network lifetime is closely tied to the energy consumption rates of sensors. Finally, the sensors sense the data about their vicinities periodically and the data generated by them are the continuous streaming data. Thus, a WSN containing N sensors can be viewed as a distributed stream system with N streaming data [1]. However, this special distributed stream system is essentially different from the traditional distributed stream system, since sensors have limited storage and processing capabilities in comparison with very powerful computers. In a sensor network there is not such a single powerful processor that is able to communicate with the other sensors and serves as the collection center. Each sensor usually transmits its data to the base station through multiple relays, and the sensors involved the data transmission consume communication energy. This implies it is more expensive to obtain the data from remote sensors from the base station than these nearby ones.

There are a few studies on skyline query in WSNs in literature [5, 12, 14, 17, 30]. For example, A simple algorithm `Skyline_merge` is described as follows [12]. A routing tree rooted at the base station is built. The leaf sensors send the skylines of their local points to their parents, while the internal sensors first calculate the skyline of its local points and received points and then send the skylines to their parents. The base station finally computes the skyline of the received points, which is the skyline of all points in the network. Huang et al. [12] dealt with a constrained skyline query problem on MANETs by devising a filter-based evaluation algorithm `DF` that is easily extended to WSN environments. In algorithm `DF`, every sensor sets its local point dominating the maximum number of local points as its own filter and the base station first broadcasts an initial filter to its children. Each sensor filters out the unlikely skyline points by its local and the received filters and broadcasts the "better" filter that dominates more local points to its children. The skyline is obtained by applying algorithm `Skyline_merge` on those non-filtered points. Chen et al. [5] proposed a maintenance algorithm that employs a virtual filtering point `MINMAX` at each sensor to filter out unlikely skyline points from transmission. Xin et al. [30] proposed two filter algorithms for skyline query. One is the single point filter algorithm `TF`, in which the expected number of points dominated by each point is first evaluated based on the given density function of data distribution. Every sensor sends such a local point that dominates the most points in it to its parent. The base station finally obtains a global filter by in-network aggregation and broadcasts the filter to the sensor network. The final skyline is obtained by applying algorithm `Skyline_merge` on non-filtered points.

Another is the grid filter algorithm GI, in which a grid partition of data space with each sensor is carried out first, which a grid cell is set to be 1 if there is at least one point located in it. The cells that are dominated by the cell being 1 are set to 0 and the other cells are set to 1. Each sensor in the tree merges the grids from its children, using the boolean ''and'' operator on the corresponding cells of different grids. Consequently, the grid index filter of entire datasets is obtained at the base station and then broadcast to every sensor for filtering points. Xin addressed the skyline maintenance as well, the filter (TF or GI) is only updated when the brought benefit of updating it excesses the cost of broadcasting it to the sensor network. Kwon et al. [14] proposed another filter-based algorithm MFT similar to algorithm DF but with a different way to choose the local filter. Liang et al. [17] proposed a new filter-based algorithm which consists of multiple points rather than a single point as the filter, in which each sensor sends part of its local skyline points chosen by a greedy algorithm to its parent and the root broadcasts the received points as the global certificate through in-network aggregation. The points that cannot pass through the certificate will be filtered out from transmission.

However, existing algorithms for skyline queries in WSNs have their own limitations. For example, the filter at each sensor in algorithms DT and MFT is only determined by the local and its parent's filters without global information, which makes their filtering abilities of filters on the sensors near to the base station very weak. Algorithm TF is based on the assumption that the density function of data is known beforehand. Such an assumption may be too restrictive in the real world. In algorithm GI, the energy overhead on the construction of the grid and the efficiency of the grid are the conflicting objectives, which both depend on the granularity of the grid. It thus is difficult to set an appropriate granularity for different datasets. The filter MINMAX may be outside the data space of points in some cases, e.g., the maximum value of one dimension is smaller than the minimum value of another dimension among all the points in 2-dimensional datasets, which means that the filter may not be able to filter out any points. On the other hand, the authors in [5] only considered the skyline maintenance without expiration of the points, which contradicts the reality that the sensed points are dynamically changed with time. Moreover, the existing algorithms mainly focus on the optimization of the total energy consumption by ignoring the maximum energy consumption among the sensors.

To process queries in sensor networks, the major optimization objective is the network lifetime, which is determined not only by the total energy consumption of all sensors but also by the maximum energy consumption among the sensors. The sensors near to the base station usually consume the most energy because they relay the data for others, and will exhaust their batteries first. Once they run out of energy, the rest of sensors will be disconnected from the base station no matter how much residual energy they left. Consequently, the base station cannot receive any data from these survival sensors, and the network is no longer functioning even if the total energy consumption per query is small. This implies that for query processing in sensor networks, only minimizing the total energy consumption is insufficient. Minimizing the maximum energy consumption among the sensors is another important optimization objective to prolong network lifetime [20]. Therefore, a desired algorithm for skyline query should not only optimize the total but also the maximum energy consumptions among the sensors. Above all, the design of energy-efficient algorithms for skyline query in WSNs poses great challenges.

1.2 Contributions

Our major contributions in this paper are as follows. We first introduce the problem of skyline evaluation and maintenance in WSNs under a sliding window environments. We then devise energy-efficient, progressive evaluation algorithms for skyline query evaluation and an incremental algorithm for skyline maintenance. The key strategy adopted is to partition the entire dataset into disjoint subsets and return the skyline points progressively through examining the subsets one by one. Also, a global filter consisting of some found skyline points in the processed subsets is used to filter out those unlikely skyline points from the rest of subsets for transmission. We finally conduct extensive experiments by simulation on both synthetic datasets and real datasets. The experimental results show that the proposed algorithms significantly outperform existing algorithms on various datasets in terms of various performance metrics.

The reminder of the paper is organized as follows. Section 2 introduces the WSN model and problem definition, followed by giving an observation which is the cornerstone of the proposed algorithms. Section 3 proposes two progressive algorithms for skyline query evaluation. Section 4 describes an energy-efficient incremental algorithm for skyline maintenance. To evaluate the performance of the proposed algorithms, extensive experiments on various datasets are conducted in Sect. 5, and the conclusions are given in Sect. 6.

2 Preliminaries

2.1 System model

We consider a WSN consisting of N stationary sensors randomly deployed in a region of interest and a base station

with unlimited energy supply serving as the gateway between the sensor network and the end users. Each sensor equipped with d different sensing devices measures d attribute values. Assume that the transmission range of each sensor is identical. Each sensor can communicate with the other sensors within its transmission range and communicate with the base station via one or multi-hop relays. The battery-powered sensors can not only sense and collect data from their vicinities but also process and transmit the data to its neighbors. To transmit a message containing l bytes of data from one sensor to another, the transmission energy consumed at the sender are $\rho_t + R*l$, and the reception energy consumed at the receiver are $\rho_r + r_e*l$, where ρ_t and ρ_r are the sum of energy overhead on handshaking, transmitting, and receiving header part of the message, R and r_e are the transmission and reception energy consumed per byte. Each sensed value is represented by 4 bytes and thus a d-dimensional point is represented by $4d$ bytes in total. We assume that the computation energy consumption on sensors can be ignored, because in practice it is several orders of magnitude less than that of the communication energy consumption. For example, the authors in [21, 24] claimed that the transmission of a bit of data consumes as much energy as executing 1,000 CPU instructions. Therefore, unless otherwise specified, we only compare the communication energy consumption of different algorithms in performance evaluation.

2.2 Problem definition

Consider a WSN as an undirected graph $G = (V, E)$ with a base station r, where V is the set of sensors and E is the set of links between the sensors or between the sensors and the base station. There is a link between two sensors or a sensor and the base station if they are within the transmission range of each other. d is the number of dimensionality of points. Assume that each sensor $v \in V$ has a set of snapshot points $P(v)$ generated during a given time interval, and $P = \cup_{u \in V} P(v)$ is the entire dataset of generated by the sensors in the network for a given time interval. *The skyline query* on the snapshot dataset P is to find a subset of P such that the points in the subset cannot be dominated by any other points in P. We refer to this subset as $SK(P)$. Without loss of generality, we assume that the value range of each dimension of a point is in $[0, +\infty)$ and the whole d-dimensional data space $DS = \{[0, +\infty), [0, +\infty), \ldots, [0, +\infty)\}$ is the union of subspaces derived by the N sensors. In case a data space D contains points with negative values, it is easy to transform the space to DS first through the coordinate transformation, and then transform the results on DS back to the results in the data space D.

Skyline maintenance is to maintain the results of skyline query dynamically within sliding window environments. The sensors sense the data from their vicinities periodically and the data generated by sensors is treated as the boundless streaming data. Thus, it is impossible to perform a skyline query on all generated points so far. Instead, a *sliding window skyline* will be considered, and only the skyline of the points generated within the current time window is computed. The length of sliding window is equal to the "lifespan" of each point. We assume that each sliding window is further divided into a number of *equal* time steps. At each time step, the points in the current window are considered in the query evaluation. The skyline maintenance at every time step t is to evaluate the skyline on set $P_t = \{p \mid t - p{\cdot}time \leq W\}$, where $p{\cdot}time$ is the generated time step of p, and W is the window length which is the lifespan of point p.

The energy-efficient skyline query processing in sensor networks can be implemented through in-network processing paradigm as follows [20, 32]. A routing tree $\mathcal{T}$ rooted at the base station r spanning all sensors is employed for such a purpose. The query processing on $\mathcal{T}$ consists of a *distribution stage*, in which the query is pushed down to each sensor along the tree paths; and a *collection stage*, in which the sensed data are routed up from children to parents and eventually to the base station through multi-hop relays.

In the following we define the *radius of a point* and an important observation proposed in [22] that is the cornerstone of the rest of the paper.

Definition 1 [13, 22] Suppose $p = (p_1, p_2, \ldots, p_d)$ is a d-dimensional point, we define $R(p) = \sqrt{\Sigma_{i=1}^{d} p_i^2}$ as the *radius* of point p, referred to as $R(p)$.

Observation 1 [22] *Let $R(p)$ and $R(q)$ be the radii of points p and q. If $R(p) \leq R(q)$, point p cannot be dominated by point q.*

3 Fixed dataset partition based algorithm

In this section, we propose a skyline evaluation algorithm based on partitioning the dataset, using a fixed partition radii. Given a d-dimensional dataset P consisting of all the points in the network, denote by $R(P)_{max} = \max\{R(p) | \ p \in P\}$ and $R(P)_{min} = \min\{R(p) | \ p \in P\}$ the maximum and the minimum radii of dataset P. The basic idea behind the proposed algorithm is to partition P into k disjoint subsets, $P_1, \ldots, P_k$, such that $R(P_i)_{max} < R(P_{i+1})_{min}$ for all $1 \leq i < k$, where k (≥ 1) is given beforehand, and the series of k radii $R(P_i)_{max}$ is determined by k. $R(P_i)_{max}$ is the *partition radius* of subset P_i. In the ith iteration, the proposed algorithm examines the points in P_i and finds a new skyline SK_i such that each point in it will not be dominated by the found skyline points so far. The skyline on set P

then is the union of newly found skylines from each subset, i.e., $SK(P) = \cup_{i=1}^{k} SK_i$.

3.1 Fixed dataset partition

$R(P)_{min}$ and $R(P)_{max}$ on dataset P can be obtained through in-network aggregation. Having these two values, the algorithm first generates a series of k ascending radii $R(P_i)_{max}$ for all $1 \leq i \leq k$. For the sake of analysis, we assume that the sequence of the partition radii is either an arithmetic or a geometric sequence.

Case one: To generate an arithmetic sequence, suppose $R(P)_{max} - R(P)_{min} = a + 2a + \cdots + ka$ and the approximate value of $a = \frac{2(R(P)_{max} - R(P)_{min})}{k(k+1)}$. Thus, $R(P_i)_{max} = R(P_{i-1})_{max} + i*a$ and $R(P_0)_{max} = R(P)_{min}$.

Case two: To generate a geometric sequence, suppose $R(P)_{max} - R(P)_{min} = q^1 + q^2 + \cdots + q^k$. Thus, $R(P_i)_{max} = R(P_{i-1})_{max} + q^i$, $1 \leq i \leq k$. According to the series of $R(P_i)_{max}$, the dataset P can be partitioned into k disjoint subsets $P_1, \ldots, P_k$ and the radii of the points in set P_i are within $(R(P_{i-1})_{max}, R(P_i)_{max}]$, $1 < i \leq k$. Particularly, the radii of the points in P_1 are within $[R(P)_{min}, R(P_1)_{max}]$.

A routing tree is first built and the algorithm then proceeds with k iterations. Denote by $SK(S)$ the skyline on set S. $\mathrm{LSK}_i(v)$ is referred to as the skyline of the points at sensor v and the received points from the children of v. $LF_i(v)$ consisting of several points is defined as *the local filter* of v in the ith iteration. Initially, $LF_1(v) = \emptyset$ and $\mathrm{LSK}_1(v) = SK(P(v))$ if v is a leaf sensor. We refer to the algorithms that partition the dataset, using the arithmetic or geometric series as algorithm `a-FDP` or `g-FDP` (*Fixed Dataset Partition*), respectively. In the ith iteration, either of the algorithms proceeds as follows.

These points at sensor v are firstly filtered out if they are dominated by any point in $LF_i(v)$. If sensor v is a leaf sensor, it sends the points in $\mathrm{LSK}(v)_i$ whose radii are no larger than $R(P_i)_{max}$ to its parent; otherwise, sensor v calculates $\mathrm{LSK}_i(v)$ of the points at sensor v and the received points, followed by transmitting the points in $\mathrm{LSK}_i(v)$ whose radii are no larger than $R(P_i)_{max}$ to its parent. In the end, the base station r calculates the skyline $\mathrm{LSK}_i(r)$ on all received points. $SK_i = \{p |\ p \in \mathrm{LSK}_i(r), \forall q \in \cup_{j=1}^{i-1} SK_j, q \not\prec p\}$ as the set of the newly found skyline points in the ith iteration. $SK(\cup_{j=1}^{i} P_j) = \cup_{j=1}^{i-1} SK_j \cup SK_i$. Finally, some points in SK_i will be chosen and broadcast back to sensors in the sensor network for filtering out unlikely skyline points in those unexamined subsets.

Denote by GSF_i the set of global skyline points broadcast in the ith iteration. Each sensor then updates its local filter with GSF_i, i.e., $LF(v)_{i+1} = LF(v)_i \cup \mathrm{GSF}_i$. Having performed the first k iterations, the skyline on dataset P is $\cup_{i=1}^{k} SK_i$. Below, we detail which points in SK_i are to be added to GSF_i in the ith iteration for algorithm `a-FDP` or `g-FDP`.

3.2 Skyline point choice for global filtering

The global filter will be used to filter out those unlikely skyline points from unprocessed subsets from transmission in future iterations. It consists of some of newly found skyline points. In the real world, it is impossible to figure out the exact number of points filtered out by the chosen skyline points before these chosen skyline points are broadcast, because there is no knowledge of data distribution in the network. Instead, *the volume of dominance region* of a point can be used to represent its filtering "gain"—the number of points is dominated by the point approximately. The *dominance region* of a point p is the region in which any point is dominated by p. Having obtained SK_i at the base station r, a simple way to update the local filter of each sensor is to broadcast all points in SK_i to each sensor. However, this naive approach will incur much more energy overhead than needed, due to the fact that the dominance regions of most found skyline points are overlapping with each other, and only very few of them cover most of the whole dominance region. On the other hand, if newly found global skyline points are not broadcast to sensors, the local filter of each sensor will become "obsolete", its filtering ability will become inefficient due to lack of the up-to-date global information. Consequently, the local filter of each sensor may not be able to filter out as many unlikely skyline points as possible, and the sensor will incur excessive energy overhead on transmitting those unlikely skyline points to its parent. In the following we propose a method to tradeoff the energy consumption between the filter broadcasting and the transmission of unlikely skyline points without filtering, based on the volume of the *efficient dominance region* of each point, where an efficient dominance region is defined as follows.

Definition 2 Assume that a skyline point $p \in SK_i$. The *efficient dominance region* of point p is defined as the subspace of the dominance region by the point in which the points have radii larger than $R(P_i)_{max}$ and are dominated by point p, but not dominated by any other point in $\bigcup_{x=1}^{i-1} SK_x$.

The intuition of choosing which skyline points from SK_i for the global filter is that efficient dominance regions of the skyline points obtained in the current iteration are in the margin space of the dominance regions of the skyline points found in previous iterations. Such an example is illustrated in Fig. 1.

Suppose that there are two skyline points p_0 and p_1 obtained in the first iteration and *Region D* is the union of dominance regions of p_0 and p_1. Points p_2 and p_3 are the found skyline points in SK_2 and $r_2 = R(P_2)_{max}$. Following algorithm `a-FDP` or `g-FDP`, after the 2nd iteration is performed, the dominance region within the fan region $S(Or_2r_2')$ will be useless for filtering purpose, since all the points in this region have already been examined. The

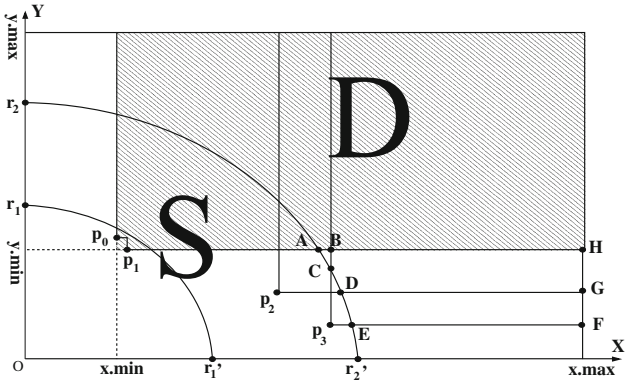


Fig. 1 An example of margin-coverage

efficient dominance regions of p_2 (*Region ACDGH*) and p_3 (*Region BCDEFH*) are actually located in the margin space of region D (between region D and X axis), because most of the dominance regions of p_2 and p_3 have been covered by *Region D* and *Region S*. From Fig. 1 we can observe that *Region BCDEFH* of p_3 covers most of *Region ACDGH* of p_2 except *Region ACB*, which implies that only broadcasting p_3 will lead to almost the same filtering gain as broadcasting both p_2 and p_3. In higher dimensional datasets, it is more complicated to calculate the volume of efficient dominance regions of the points. We extend our analysis to a more general case by developing a greedy approach, which is described as follows.

Assume that algorithm `a-FDP` or `g-FDP` performs iteration i. Denote by $\mathrm{EDR}(p)_j$ the approximate volume of the efficient dominance region of point p at the jth dimension. Let GSF_i be the set of chosen skyline points to broadcast after the ith iteration. Given a set P of d dimensional points, the point $MAX = (max_1, \ldots, max_d)$ is a *virtual point*, where max_j is the maximum value at the jth dimension of the points in P and the point $MinSK(i) = (minSK(i)_1, \ldots, minSK(i)_d)$ is another *virtual point*, where $minSK(i)_j$ is the minimum value at the jth dimension of all found skyline points in the first i iterations. Let $margin(p)_j = minSK(i-1)_j - p_j$, which represents how far point p from the dominance regions of all found skyline points in previous iterations at the jth dimension. The approximate volume of the efficient dominance region of point $p = (p_1, p_2, \ldots, p_d)$ in SK_i at the jth dimension is $\mathrm{EDR}(p)_j = margin(p)_j * (\prod_{k=1,k\neq j}^{d}(max_k - p_k) - \prod_{k=1,k\neq j}^{d}(R(P_i)_{max} - p_k))$, where $R(P_i)_{max}$ is the partition radius of subset P_i of P. For each dimension j, the point p in SK_i with the maximum value of $\mathrm{EDR}(p)_j$ and $\mathrm{EDR}(p)_j > 0$ is chosen and added to GSF_i if $p \notin \mathrm{GSF}_i$. If there are multiple points with the maximum values of EDR, the point with the minimum radius will be chosen. In the end, at most d skyline points (one point chosen in each dimension) are broadcast into the sensor network to update the local filters after the ith iteration. The virtual point MAX can be obtained using in-network processing within the first iteration. In the first iteration, $\mathrm{GSF}_1 = \{p|\ p \in SK_1, \exists j \in 1, 2, \ldots, d\}, p_j = \min\{q_j|\ q \in SK_1\}\}$.

3.3 Correctness of algorithms `a-FDP` and `g-FDP`

The rest is to show that $SK(P)$ obtained after k iterations by algorithm `a-FDP` and `g-FDP` is the skyline on set P by the following theorem, where k is the number of subsets partitioned.

Theorem 1 *For a dataset P, $SK(P)$ is referred to as the skyline on P, and $SK(P) = \cup_{i=1}^{k} SK_i$, where SK_i is the new skyline points delivered by algorithm* `a-FDP` *or* `g-FDP` *in the ith iteration, $1 \leq i \leq k$.*

Proof Assume that there is a point p and its radius $R(p) \in (R(P_{i-1})_{max}, R(P_i)_{max}]$. We show the claim by proving that if $p \notin SK_i, p$ must be dominated by the other points; otherwise, p cannot be dominated by any other points in P. Clearly, only the points whose radii are in between $R(P_{i-1})_{max}$ and $R(P_i)_{max}$ can possibly be added to SK_i, because in the ith iteration, algorithm `a-FDP` or `g-FDP` examines all points in subset P_i, whose radii range from $R(P_{i-1})_{max}$ to $R(P_i)_{max}$. It is obvious that point $p \notin SK_i$ is dominated by other points. Otherwise, it will be relayed to the base station and added to SK_i. For a point $p \in SK_i$, we show that point p is not dominated by the other points into three cases. □

Case 1. *p is not dominated by any point in $\cup_{j=1}^{i-1} P_j$.* If there is a point $q' \in \cup_{j=1}^{i-1} P_j$ dominating point p, there must be a point $q \in \cup_{j=1}^{i-1} SK_j$ such that $q \prec q'$ and $q \prec p$, or $q = q'$, this contradicts the fact that $SK_i = \{p|\ p \in \mathrm{LSK}_i(r), q \not\prec p, q \in \cup_{j=1}^{i-1} SK_j\}$.

Case 2. *p is not dominated by any point in P_i.* Assume that point p is relayed to sensor v in the ith iteration. If there is a point q at sensor v with $q \in P_i$ and $q \prec p, p$ is impossible to be added to $\mathrm{LSK}_i(v)$ and transmitted to the parent of v, otherwise this contradicts the fact that $p \in SK_i$.

Case 3. *p is not dominated by any point in $\cup_{j=i+1}^{k} P_j$.* The range of radii of the points in subset P_i is from $R(P_{i-1})_{max}$ to $R(P_i)_{max}$, the points in $\cup_{j=i+1}^{k} P_j$ thus have larger radii than that of point p. From Observation 1, it is obvious that p cannot be dominated by any point in $\cup_{j=i+1}^{k} P_j$.

In summary, point $p \in SK_i$ is not dominated by any point in P. Therefore, $SK(P) = \cup_{i=1}^{k} SK_i$ contains all the skyline points in P.

`Algorithms` 1 and 2 describe the algorithm of determining the global filter in the ith iteration and algorithm `a-FDP` (`g-FDP`).

Algorithm 1: `Algorithm GSF`(i, SK_i)

begin
- $\text{GSF}_i = \emptyset$;
- **if** $i = 1$ **then**
  - **for** *each dimension* j $(1 \le j \le d)$ **do**
    - **if** $(p_j = \min\{q_j \mid q \in SK_i\})$ *and* $(p \notin GSF_i)$ **then**
      - $\text{GSF}_i \leftarrow \text{GSF}_i \cup \{p\}$;
  - **else**
    - **for** *each dimension* j $(1 \le j \le d)$ **do**
      - **if** $(EDR(p)_j = \max\{EDR(q)_j \mid q \in SK_i\})$ *and* $(EDR(p)_j > 0)$ *and* $(p \notin GSF_i)$ **then**
        - $\text{GSF}_i \leftarrow \text{GSF}_i \cup \{p\}$;
- Update the virtual point $MinSK(i)$ and broadcast GSF_i;

end

Algorithm 2: `Algorithm` **a-FDP** (**g-FDP**)(k, P, V, E)

Input: The data set P stored in sensor nodes.

Output: The skyline $SK(P)$ of the data set P.

begin
- $SK(P) \leftarrow \emptyset$;
- Generate arithmetic (geometric) series $R(P_i)_{max}, \forall i \in [1, k]$;
- Broadcast the series $R(P)_{max}, i \in [1, k]$ to the network;
- **for** $i \leftarrow 1$ *to* k **do**
  - **for** *each sensor* v **do**
    - Filter out the points at v using $LF_i(v)$;
    - **if** v *is a leaf sensor* **then**
      - Calculate $\text{LSK}_i(v)$;
      - Send the set $\{p \mid p \in \text{LSK}_i(v), R(p) \le R(P_i)_{max}\}$ to its parent;
    - **else**
      - Receive the points from the children;
      - Calculate $\text{LSK}_i(v)$;
      - Send the set $\{p \mid p \in \text{LSK}_i(v), R(p) \le R(P_i)_{max}\}$ to its parent;
  - The base station r calculates $\text{LSK}_i(r)$;
  - $SK_i \leftarrow \{p \mid p \in \text{LSK}_i(r), \forall q \in \cup_{j=1}^{i-1} SK_j, q \not\prec p\}$;
  - $SK(P) \leftarrow SK(P) \cup SK_i$;
  - **Call** `Algorithm` 1 `GSF`(i, SK_i);
  - **for** *each sensor* v **do**
    - $LF_{i+1}(v) \leftarrow LF_i(v) \cup \text{GSF}_i$;
- **Return** $SK(P)$;

end

4 Dynamical dataset partition based algorithm

In this section we deal with dynamic partition of a given dataset. Although later experiments indicate that algorithms a-FDP and g-FDP outperforms the *skyline_merge* algorithm, in terms of the total energy consumption and the maximum energy consumption among the sensors, they do suffer the following shortcomings.

Firstly, they take one extra preprocessing iteration on the routing tree to obtain the maximum and the minimum radii of the points in P without any skyline points delivered in the iteration. Secondly, in a dataset with skewed data distribution, it is unavoidable that some disjoint subsets derived based on the given partition radii are empty. One such scenario is that if the chosen value of k is too large, P is partitioned into many small disjoint subsets. Even if $P_i = \emptyset$, algorithm a-FDP or g-FDP still executes the ith iteration, which leads to the energy consumption without any gain. Finally, choosing an appropriate k is very difficult, because the performance of algorithms a-FDP and g-FDP depends heavily on different values of k based on different datasets, which will be confirmed later in the section of performance evaluation. To overcome these shortcomings by the fixed dataset partition, in the following we propose an algorithm DDP (*Dynamic Dataset Partition*), which partitions the dataset dynamically.

4.1 Dynamical dataset partition

The proposed algorithm DDP will follow the same spirit of algorithms a-FDP and g-FDP, that is, the data set P is partitioned into several disjoint subsets and the data (or points) in each subset is bounded by a radius. The difference between DDP and a-FDG (or g-FDP) is that the number of subsets k is not given in advance, the partition data radii of each subset are generated dynamically, and they will be determined by the value distribution of data in P.

Specifically, in each iteration, each sensor dynamically decides the upper bound radius of the points it can send to its parent. When the points are sent to the base station in the end, the partition data radius of the subset of this iteration is determined, and the points that are included in the subset have radii between the previous and the current partition data radii. The detailed algorithm DDP is depicted as follows.

Let $UR_i(v)$ be the *upper bound* on the partition data radius of sensor v in the ith iteration. If v is a leaf sensor, $UR_i(v)$ is the maximum radius among all points forwarded by v; otherwise, $UR_i(v)$ is calculated as follows.

Assume that v has d_v children, $u_1, \ldots, u_{d_v}$. Each child u_j sends all points in it whose radii is smaller than $UR_i(u_j)$ to sensor v, $1 \le j \le d_v$. Sensor v calculates $\mathrm{LSK}_i(v)$ that is the skyline of the points received from its children and the points generated by itself. Then, $UR(v)_i = \min\{R(\mathrm{LSK}_i(v))_{max}, UR_i(u_1), \ldots, UR_i(u_{d_v})\}$. Obviously, $UR_i(v) \le UR_i(u)$ if u is a descendant sensor of v. Thus, the upper bound $UR_i(r)$ on the partition data radius of the base station r is smaller than that of any sensor in the network, i.e., $\forall v \in V, UR_i(r) \le UR_i(v)$. $UR_i(r)$ will be the *partition data radius* of the ith partitioned subset P_i, i.e., the radii of all the points in P_i are no greater than $UR_i(r)$. The dataset P is dynamically partitioned by the partition data radius $UR_i(r)$. Suppose that the points in each sensor v are sorted in increasing order of their radii. Assuming that algorithm DDP has performed the first $(i-1)$th iterations already, it now proceeds with the ith iteration.

If sensor v is a leaf sensor, it transmits all the points in $\mathrm{LSK}_i(v)$ to its parent; otherwise, it first calculates $\mathrm{LSK}_i(v)$ and the upper bound on the transmission data radius of sensor v, $UR_i(v)$, then transmits the points in $\mathrm{LSK}_i(v)$ whose radii are no greater than $UR_i(v)$ to its parent. Having received the points from all of its children, the base station r calculates $\mathrm{LSK}_i(r)$ and $UR_i(r)$. The newly found skyline in the ith iteration is $SK_i = \{p |\ p \in \mathrm{LSK}_i(r), R(p) \le UR_i(r), \forall q \in \cup_{j=1}^{i-1} SK_j, q \not\prec p\}$. Some of the skyline points in SK_i later will be broadcast to the sensors again as the global filter to update the local filter of each sensor v, i.e., $LF_{i+1}(v) = LF_i(v) \cup \mathrm{GSF}_i$, where GSF_i is the set of points broadcast in the ith iteration.

In algorithm DDP, a leaf sensor transmits all the points to its parent. However, transmitting all points at each leaf sensor will incur excessive energy consumption. Consider an extreme case, assume that all the points at leaf sensor v are only dominated by a point at another sensor u. Let the base station r be the *least common ancestor* of sensors v and u in the routing tree. This implies that all the points at sensor v will not be filtered out until they are relayed to the base station, which consumes much unnecessary energy. To this end, we introduce a fixed parameter α for limiting the number of points sent by leaf sensors in each iteration as follows.

Recall that the points in $\mathrm{LSK}_i(v)$ at sensor v are sorted in increasing order of their radii. If sensor v is a leaf sensor, it only transmits the first $\lceil(\alpha * |\mathrm{LSK}_1(v)|)\rceil$ points in $\mathrm{LSK}_i(v)$ to its parent in the ith iteration, where α is a constant with $0 < \alpha \le 1$. If $\alpha = 1$, all skyline points at leaf sensors will be transmitted. The algorithm that partitions the dataset dynamically with parameter α is referred to as algorithm α-DDP.

The use of parameter α can reduce the upper bound on the partition data radius of leaf sensors, which also reduces the partition data radius of each subset, thereby increasing

the number of iterations k. The more subsets are partitioned, the more unlikely skyline points will be filtered out. Compared to its special case algorithm 1-DDP where $\alpha = 1$, although algorithm α-DDP takes more iterations, it reduces the energy consumption of sensors from data transmission, which can be verified in the later performance evaluation. By using parameter α, SK_i may be empty if all the transmitted points in the current iteration are all dominated by the found skyline points. In this case, algorithm skyline_merge will be applied in a final iteration for finding the remaining skyline points and then the number of iterations k is determined. The skyline of set P is $SK(P) = \cup_{i=1}^{k} SK_i$. Algorithm 3 is the pseudo-code of algorithm α-DDP.

Theorem 2 *Given a dataset P, the skyline $SK(P)$ of P is $SK(P) = \cup_{i=1}^{k} SK_i$, where SK_i is the new skyline points delivered by algorithm α-DDP in the ith iteration.*

Proof Following the construction of upper bound on the maximum partition data radii, we note that only the points whose radii are ranged from $UR(r)_{i-1}$ to $UR(r)_i$ are likely to be in SK_i, which implies that the subset P_i of P contains only the points with radii between $UR_{i-1}(r)$ and $UR_i(r)$. Once $SK_i = \emptyset$, this implies that there are no skyline points with radii between $UR_{i-1}(r)$ and $UR_i(r)$. The skyline_merge algorithm is then applied to the tree, and it will return skyline points with radii between $UR_i(r)$ and $R(P)_{max}$, where $R(P)_{max}$ is the maximum of the radii of the points in P. □

Algorithm 3: Algorithm α-DDP(P, V, E, α)

Input: The data set P stored in sensor nodes.

Output: The skyline $SK(P)$ of the data set P.

begin

$i \leftarrow 1$; $SK(P) \leftarrow \emptyset$;

repeat

for *each sensor v* **do**

Filter out some unlikely skyline points at v, using $LF(v)_i$;

if *sensor v is a leaf sensor* **then**

calculate $\mathrm{LSK}_i(v)$;

sort the points in $\mathrm{LSK}_i(v)$ in increasing order of their radii;

send the first $\alpha|\mathrm{LSK}_i(v)|$ points with smaller radii to its parent;

else

receive the points from the children;

calculate $\mathrm{LSK}_i(v)$ and $UR_i(v)$;

send the set $\{p \mid p \in \mathrm{LSK}_i(v), R(P) \leq UR_i(v)\}$ to its parent;

the base station r calculates $\mathrm{LSK}_i(r)$;

$SK_i \leftarrow \{p \mid p \in \mathrm{LSK}_i(r), R(p) \leq UR_i(r), \forall q \in \cup_{j=1}^{i-1} SK_j, q \not\prec p\}$;

Call algorithm GSF(i, SK_i);

for *each sensor v* **do**

$LF_{i+1}(v) \leftarrow LF_i(v) \cup \mathrm{GSF}_i$;

$SK(P) \leftarrow SK(P) \cup SK_i$;

$i \leftarrow i + 1$;

until $SK_i = \emptyset$;

SK_{i+1} is obtained by applying the Skyline_merge algorithm on the rest of points in sensors;

$SK(P) \leftarrow SK(P) \cup SK_{i+1}$;

Return $SK(P)$;

end

4.2 Correctness of algorithm α-DDP

We now show that set $SK(P)$ delivered by algorithm α-DDP is the skyline on set P by the following theorem.

Algorithm α-DDP partitions the set P into k subsets and the radii range of the points in each subset P_i is within $(UR_{i-1}(r), UR_i(r)]$ when $1 \leq i \leq k-1$. The radii of the points in the last subset P_k are ranged from $UR_{k-1}(r)$ to

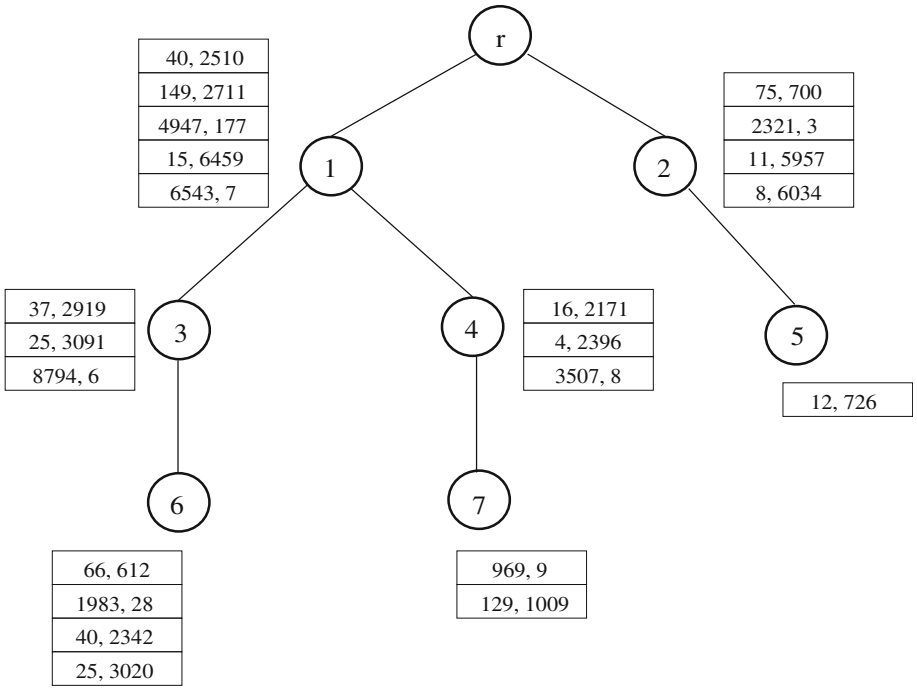


Fig. 2 The points in the sensor network

$R(P)_{max}$. The rest arguments are similar to the ones in Theorem 1. For the points not being in SK_i, obviously they are dominated by the other points. Otherwise, they will be sent to the base station and added into SK_i. In the ith iteration, only these points in P_i whose radii are between $UR_{i-1}(r)$ and $UR_i(r)$ can potentially be added to SK_i. If a point p is dominated by any point in $\cup_{j=1}^{i-1} P_j$, it must be dominated by a point in $\cup_{j=1}^{i-1} SK_i$ and it cannot be possible to be added to SK_i. If a point p is dominated by any point q in P_i, p will be removed if p and q (or another point dominating q in P_i) are sent to the same sensor node or to the base station. Thus, it is impossible that p will be added to SK_i. And for all other points in $\cup_{j=i+1}^{k} P_j$, their radii are larger than that of point p, they cannot dominate point p. In summary, all the points in SK_i are the skyline points in P. Therefore, we conclude $SK(P) = \cup_{i=1}^{k} SK_i$.

4.3 An example

In this subsection we use an example to illustrate the major steps of algorithm α-DDP. Suppose that the routing tree has 7 sensors including the base station. α is set to be 0.5. As shown in Fig. 2, each sensor contains several skyline points arranged in increasing order of radii. Figures 3, 4 and 5 illustrate the details of the first two iterations of algorithm α-DDP. Within the first iteration, every leaf sensor sends the first half points with smaller radii from its local skyline points. Sensor 5 sends point (12, 726) to sensor 2, sensor 6 sends points (66, 612) and (1983, 28) to sensor 3, and sensor 7 sends point (969, 9) to sensor 4. We thus have $UR_1(5) = 726.1$, $UR_1(6) = 1983.2$, and $UR_1(7) = 969.04$ because the upper bound on the transmission data radius of a leaf sensor is the maximum radius the points sent by it.

Having received the points from their children, it can be seen that $R(\text{LSK}_1(3))_{max} = 8794$ by point (8794, 6), which is the maximum radius of points in $\text{LSK}_1(3)$, and then $UR_1(3) = \min\{R(\text{LSK}_1(3))_{max}, UR_1(6)\} = \min\{8794, 1983.2\} = 1983.2$. Therefore, sensor 3 only sends (66, 612) and (1983, 28) to sensor 1, since their radii are smaller than $UR_1(3)$ (=1983.2). Similarly, $R(\text{LSK}_1(4))_{max} = 3507$ by point (3507, 7) and $UR_1(4) = \min\{3507, 969.04\} = 969.04$. Thus, sensor 4 sends (969, 9) to sensor 1. Having received the points, sensor 1 calculates $\text{LSK}_1(1)$ and removes points (1983, 28) and (4947, 177) dominated by point (969, 9). The $R(\text{LSK}_1(1))_{max}$ is 6543 of (6543, 7) and then $UR_1(1) = \min\{R(\text{LSK}_1(1))_{max}, UR_1(3), UR_1(4)\} = 969.04$. Therefore, sensor 1 only sends points (66, 612) and (969, 9) to the base station r. Similarly sensor 2 sends (12, 726) to the base station r. Having received the 3 points, $\text{LSK}_1(r)$ is {(66, 612), (12, 726), (969, 9)} and $R(\text{LSK}_1(r))_{max}$ is 969.04. In the end $UR_1(r) = \{969.04, 969.04, 726.1\} = 726.1$. Thus, points (66, 612) and (12, 726) are regarded as the skyline points in the first iteration, i.e., $SK_1 = \{(66, 612), (12, 726)\}$.

Having obtained SK_1, we are ready to calculate GSF_1. In the first iteration, the point which has the minimum value at each dimension is added to GSF_1 so that points (66, 612) and (12, 726) are chosen for the global filter and broadcast to the network. Each sensor then removes the points dominated by the points in the global filter. The detail of the first iteration is illustrated in Figs. 3 and 4.

In the second iteration, sensors 2, 3, and 4 do not receive any points from their children, meaning that the upper bounds on the transmission data radii are the maximum radii of their skyline points, i.e., $UR_2(2) = 6034$, $UR_2(3) = 8794$, and $UR_2(4) = 3507$. Thus, sensors 2, 3, and 4 send all the remaining skyline points to their parents. Having received points, sensor 1 calculates $\mathrm{LSK}_2(1)$, and $R(\mathrm{LSK}_2(1))_{max} = 8794$ by point (8794, 6). Then $UR_2(1) = \min\{8794, 3507, 8794\} = 3507$. Sensor 1 only sends points (4, 2396) and (3507, 7) to the base station r, and then the base station calculates $\mathrm{LSK}_2(r)$ and removes the dominated points (3507, 7), (11,5957), and (8,6034), where $R(\mathrm{LSK}_2(r))_{max} = 2396$ by point (4, 2396). $UR_2(r) = \min\{2396, 3507, 6034\} = 2396$. Therefore, $SK_2 = \{(969, 9), (2321, 3), (4, 2396)\}$.

GSF_2 is then calculated as follows. The virtual point $MAX = (8794, 6459)$ is obtained by piggyback to the base station during the first iteration transmission. Another

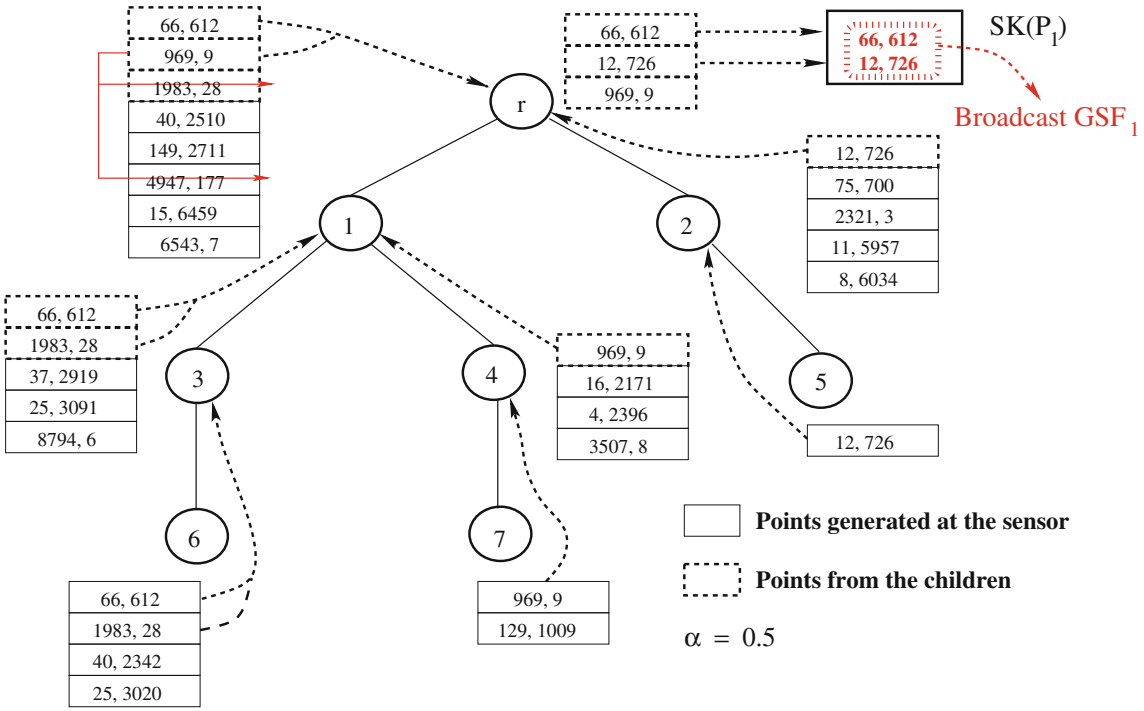
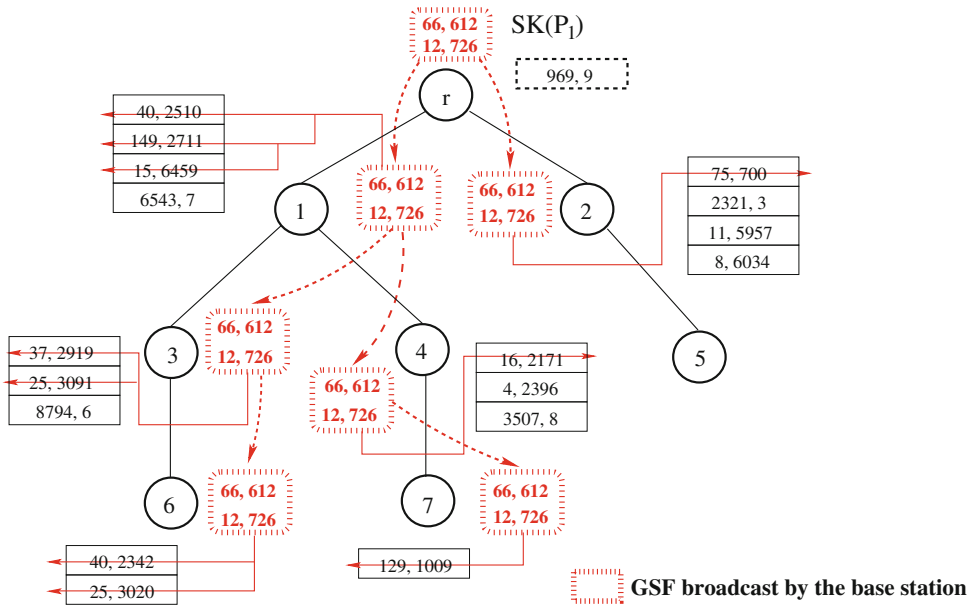


Fig. 3 The execution of the first iteration in algorithm α -DDP

Fig. 4 The broadcasting of the first iteration in algorithm α -DDP



virtual point is $MinSK = (12, 612)$, where each value at each dimension is the minimum value of the found skyline points (66,612) and (12,726). At the first dimension, $EDR((969, 9))_1 = (66 - 969) * ((9999 - 9) - (2396 - 9)) < 0$. Similarly, $EDR((2321, 3))_1 < 0$ but $EDR((4, 2396)) > 0$. Thus, point (4, 2396) is added to GSF_2. At the second dimension, $EDR((4, 2396))_2 < 0$, while $EDR((969, 9))_2 = 4.58 \times 10^6$ and $EDR((2321, 3))_2 = 4.63 \times 10^6$. Therefore, point (2321, 3) is added to GSF_2 and consequently $GSF_2 = \{(2321, 3), (4, 2396)\}$ is broadcast to the network. All the remaining points in sensor 1 are dominated by the global filter and removed. Since all the points in the network have been examined, the algorithm terminates. The process of the second iteration is shown in Fig. 5.

5 Skyline maintenance algorithm

In previous sections we deal with skyline query evaluation based on a snapshot dataset. Once the initial skyline is found, the rest is to monitor the skyline continuously over time. To this end, a naive approach is to employ algorithm α-DDP on the updated datasets to find new skylines. However, it is noted that most recent found skyline points are still the skyline points in the near future due to minor data changes in the dataset, i.e., there are very few new point generations and expirations. Consequently, instead of finding the new skyline from scratch, what we need is to maintain the skyline incrementally. Specifically, in this section we propose an algorithm MSM (*M*onitoring *S*kyline *M*aintenance) for skyline maintenance in a sliding window environment. Unlike previous skyline maintenance studies in centralized databases by assuming that there is only either an insertion or a deletion (an expiration) at each moment, the skyline maintenance in sensor networks deals with multiple independent updating at sensors concurrently. Thus, the centralized skyline maintenance approaches are not applicable to the skyline maintenance in distributive WSNs. In the following we propose a novel algorithm for such a purpose.

5.1 Algorithm overview

The idea is to maintain the skyline incrementally. That is, only potential skyline points in sensors will be sent during the maintenance period, where potential skyline points include the points generated at the current time step or the existing points dominated by the skyline points to be expired at the current time step. We assume that the data updates and skyline maintenance can be performed within one time step.

Recall that W is the lifespan of a point, which is also the length of the sliding window. Assume that the initial skyline of the points in the network can be obtained by algorithm α-DDP. If a point p is generated at time step t, $p.time = t$. The set of "valid" points at time step t is referred to as P_t, which contains non-expired points in P_{t-1} and points generated at time step t. Let $P(v)_t$ be the set of points at sensor v at time step t, which is a subset of P_t. The local filter of sensor v at time step t is denoted by $LF(v)_t$.

The proposed algorithm consists of two phases: Phase one is to update the new skyline points by traversing the routing tree in a bottom-up fashion. The base station then obtains all new skyline points, which is the skyline of

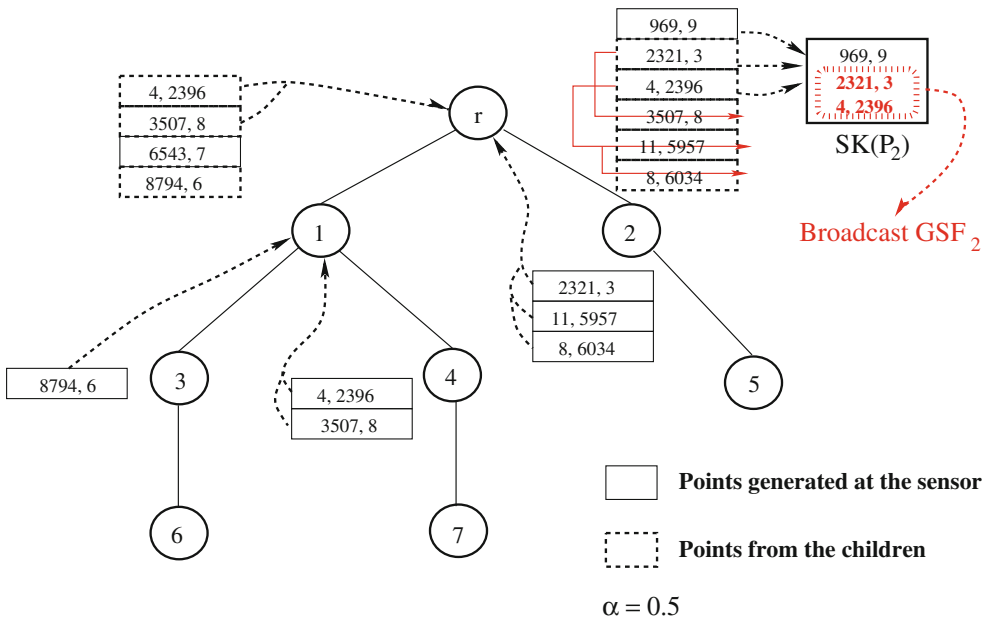


Fig. 5 The execution of the second iteration in algorithm α -DDP

newly found skyline points and the currently valid skyline points (at the base station). Phase two is to update the global filter. The base station determines which skyline points to be included in the global filter, it also decides which of these skyline points to be broadcast at the current time step in order to update the local filter of each sensor. Denote by GSF_t the set of points broadcast at time step t. Once GSF_t is broadcast, every sensor v updates $LF(v)_t$ by adding the points in GSF_t into the filter. The detailed algorithm is depicted as follows.

5.2 Skyline point updating

At time step t, each sensor v first updates its dataset $P(v)_t$ by removing expired points from it and adding newly generated points into it. It then removes the expired points from its local filter. If a point $p \in P(v)_t$ is dominated by another point $q \in P(v)_t$ or $LF(v)_t$ and $q.time \geq p.time$, p is safely removed, because q will expire later than p, and it is impossible that p will become a skyline point in the future. The `skyline_merge` algorithm is then applied on the set of remaining points which are not dominated by the points in $LF(v)_t$ in order to find the new skyline points.

Having obtained the skyline $NewSK_t$ by algorithm `skyline_merge`, the skyline on the union of $NewSK_t$ and the set of non-expired points in $SK(P_{t-1})$ is the skyline of the dataset at time step t. The base station finally determines whether to broadcast part of the skyline points obtained to update the local filter of each sensor since its last broadcast. Furthermore, which skyline points should be chosen for broadcasting is another important issue, in the following we address this issue by proposing two heuristics.

5.3 Broadcasting

We first address which skyline points will be chosen for the global filter to avoid excessive energy consumption of broadcasting all skyline points. In terms of choosing skyline points, two factors should be considered: one is that the longer the remaining lifespan a point, the more points it can potentially filter out; another is that the larger the volume of efficient dominance region of a point, the more points it can filter out.

The aforementioned approach in the proposed evaluation algorithms to determine the global filter requires the maximum value of each dimension of all points and the partition radius of each disjoint subset. However, within sliding window environments, the information about the points at time step t is not given in advance, thus a simple function $SEDR(p, t)_j$ is used to evaluate the volume of efficient dominance region of point p at the jth dimension at time step t, whose calculation is as follows.

Denote by $GSF_{t'}$ the set of points broadcast last time at time step t' and $GSF_{t'}(t)$ the set of non-expired points in $GSF_{t'}$ at time step t, i.e., $GSF_{t'}(t) = \{p |\ p \in GSF_{t'}, p.time > t - w\}$. Point $(minSK(t)_1, \ldots, minSK(t)_d)$ is a d-dimensional virtual point, where $minSK(t)_i$ is the minimum value at the ith dimension among all the points in $GSF_{t'}(t)$. Let $margin(p, t)_j$ be the distance from point p to the region being dominated by the skyline points in $GSF_{t'}(t)$ at the jth dimension. If $minSK(t)_j > p_j$, $margin(p, t)_j = minSK(t)_j - p_j$; otherwise, $margin(p, t)_i = \infty$.$SEDR(p, t)_j = margin(p, t)_j * \prod_{k=1, k\neq j}^{d}(p_k)$. A point p with smaller $SEDR(p, t)_i$ is able to filter out more points.

The heuristic of choosing points for GSF_t thus is as follows. At time step t, for every point p in set $SK(P_t) - GSF_{t'}(t)$, the weight $W(p)_j = SEDR(p, t)_j/(w - t + p.time)$ at each dimension j is first calculated and then point p is added to GSF_t if $W(p)_j$ is the minimum among the values of all the points and $p \notin GSF_t, 1 \leq j \leq d$, where $w - t + p.time$ is the remaining lifespan of p. If there are multiple points with the minimum weight, the one with the minimum radius will be chosen. Finally, GSF_t containing up to d points will be broadcast to the network.

Given GSF_t, a naive solution is to broadcast GSF_t at every time step. However, this method leads to excessive energy consumption on broadcasting. A smart solution is to broadcast GSF_t when the benefit brought through filtering out more points by GSF_t exceeds the overhead on broadcasting. We then address when the base station should broadcast GSF_t to the sensor network. At time step t, if $GSF_{t'}(t) = \emptyset$, r broadcasts GSF_t into the network; otherwise, the trigger for broadcasting is determined by whether the gain of filtering out more points using the updated local filters outweighs the overhead on updating the global filter.

We analyze the gain by broadcasting GSF_t first. Define $\overline{T_t} = \Sigma_{p\in GSF_t}(W - t + p.time)/|GSF_t|$.$GSF_t$ is expected to be part of the local filter within the time interval $(t, t + \overline{T_t}]$ if it is broadcast at time step t. Let S_t be the set of the points received by r, which are dominated by a point in GSF_{t-1} at time step t. If GSF_{t-1} was broadcast at time step $t - 1$, all the points in S_t would be filtered out immediately at their generators and would not be relayed to r. Thus, the total amount of energy saving is at least $\Sigma_{p\in S_t} h(p, v)4(d + 2)(R + r_e)$, where $h(p, v)$ is the number of hops between r and generator sensor v of point p. The size of a d-dimensional point plus its generator ID and its generation time is $4*(d + 2)$ bytes. We use this energy saving to predict that the amounts of saving is similar in the future if r broadcasts GSF_t at time step t. Thus, the total amount of energy savings, $E_{save}(t)$ is at least the sum of the saved energy during $(t, t + \overline{T_t}]$. Thus,

$$E_{save}(t) \geq \Sigma_{p \in S_t} h(p, v) * 4(d+2) * (R + r_e)\overline{T_t}. \quad (1)$$

Inequality (1) means that broadcasting GSF_t will save $\Sigma_{p \in S_t} h(p, v) * 4(d+2) * (R + r_e)$ amounts of energy at each time step in the period from $t+1$ to $t + \overline{T_t}$.

Next, we calculate the overhead on updating GSF_t at time step t, which includes: (i) the energy consumption overhead on broadcasting GSF_t to the sensor network; and (ii) the unpaid energy saving by the global filter broadcast at time step t' if $t < \overline{T_{t'}} + t'$. If the base station r broadcasts GSF_t into the sensor network, every sensor will receive a message containing GSF_t from its parent and then send the same messages to its children so that the energy overhead on broadcasting GSF_t at each sensor is $d*4d*(R + r_e) + (\rho_t + \rho_r)$. Thus, the total amount of energy overhead on broadcasting to the network is smaller than $(d*4d*(R + r_e) + (\rho_t + \rho_r))*N$. On the other hand, $GSF_{t'}$ is expected to be part of the local filter of each sensor from t' to $t' + \overline{T_{t'}}$, where $\overline{T_{t'}}$ is the average remaining lifespan of all points in $GSF_{t'}$. However, if r broadcasts GSF_t at time step t during $(t', t' + \overline{T_{t'}}]$, this means that the last global filter broadcast at time step t' does not deliver its promised energy saving of energy for the period from $t+1$ to $t' + \overline{T_{t'}}$. Thus, the amounts of unpaid energy is $\Sigma_{p \in S_{t'}} h(p, v) 4(d+2)(R + r_e) * (\overline{T_{t'}} + t' - t)$, 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 updating GSF_t, $Cost(t)$, is

$$\begin{aligned} Cost(t) \leq\ & (4d^2(R + r_e) + (\rho_t + \rho_r))N \\ & + \Sigma_{p \in S_{t'}} h(p, v) 4(d+2)(R + r_e)(\overline{T_{t'}} + t' - t). \end{aligned} \quad (2)$$

Combined inequalities (1) and (2), when $t < \overline{T_{t'}} + t'$, if the gain of filtering out more points exceeds the updating of the global filter (including the cost of broadcasting of the filter and the unpaid energy saving of the filter broadcast at t'), GSF_t will be broadcast as the global filter. Thus, we have

$$\begin{aligned} \Sigma_{p \in S_t} h(p, v) 4(d+2)(R + r_e)\overline{T_t} \geq\ & (4d^2(R + r_e) + (\rho_t + \rho_r))N \\ & + \Sigma_{p \in S_{t'}} h(p, v) 4(d+2)(R + r_e)(\overline{T_{t'}} + t' - t), \end{aligned} \quad (3)$$

Otherwise, the triggering of updating the filter is determined by whether the gain by updating the filter exceeds the cost of broadcasting the filter, that is,

$$\begin{aligned} \Sigma_{p \in S_t} h(p, v) 4(d+2)(R + r_e)\overline{T_t} \geq\ & (4d^2(R + r_e) \\ & + (\rho_t + \rho_r))N \end{aligned} \quad (4)$$

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

6 Performance study

In this section we evaluate the proposed algorithms against existing algorithms in terms of the total energy consumption of all sensors and the maximum energy consumption among the sensors. The lifetime of sensor networks here is defined as the duration from the moment when network receives the first skyline query to the moment when the first sensor runs out of its energy. Therefore, the less the total and the maximum energies consumed among the sensors for answering queries, the longer the network lifetime will be.

6.1 Experiment setting

We assume that the sensor network is deployed for monitoring a 100 m × 100 m region of interest. Within the region, 500 sensors are randomly deployed by the `NS-2` simulator [26] and the base station is located at the center of the square. There is a link between two sensors if they are within the transmission range of each other. We further assume that all sensors have the same transmission range (10 meters in this paper). As mentioned previously, the energy overhead on communication dominates the various energy consumption of a sensor and consequently we consider only the energy consumption on wireless communication in our experiments. It is supposed that the energy overhead on transmitting and receiving a header and the handshaking of a message are $\rho_t = 0.4608$ mJ and $\rho_r = 0.1152$ mJ, while the energy consumption of transmitting and receiving per byte are $R = 0.0144$ mJ and $r_e = 0.00576$ mJ, respectively, following the parameters given in a commercial sensor MICA2 [10]. In our experiments, we use the synthetic datasets with correlated, independent and anti-correlated distributions. In each dataset 10^6 points are generated following the distribution, and each sensor is assigned 2,000 points randomly. We also use the real sensing dataset obtained by Intel Lab at *UC Berkeley* [11], which is a data collection of 54 sensors. To assign the data to a network of 500 sensors in our setting, we partition the sensed sequence generated by each sensor into 10 consecutive segments and assign each segment to a sensor in our experiments. The data consists of four attributes as the 4-dimensional dataset: temperature, humidity, light, and voltage traces. We use any 2 or 3 dimensional combinations of this 4-dimensional dataset to generate 2 and 3 dimensional datasets.

6.2 Performance of fixed dataset partition algorithms

We first evaluate the performance of algorithms `a-FDP` and `g-FDP` by varying k from 2 to 6. Figure 6(a)–(f) show

the curves of the total energy consumption and the maximum energy consumption among the sensors by algorithms a-FDP and g-FDP on 2, 3, and 4 dimensional independent datasets. It can be seen that the energy consumption of algorithm a-FDP or g-FDP decreases with the increase of the value of k when k is small. With the further increase of the value of k, the performance of both algorithms becomes worse. It implies that partitioning the dataset P into too many subsets may not help reduce the energy consumption. Furthermore, the performance of algorithms a-FDP and g-FDP varies greatly with different values of k, and thus an appropriate k plays a key role in the performance. The best performance of the total energy consumption and the maximum energy consumption among the sensors by algorithm g-FDP ($k = 3$) is always better than that by algorithm a-FDP ($k = 4$). This implies that algorithm g-FDP outperforms algorithm a-FDP.

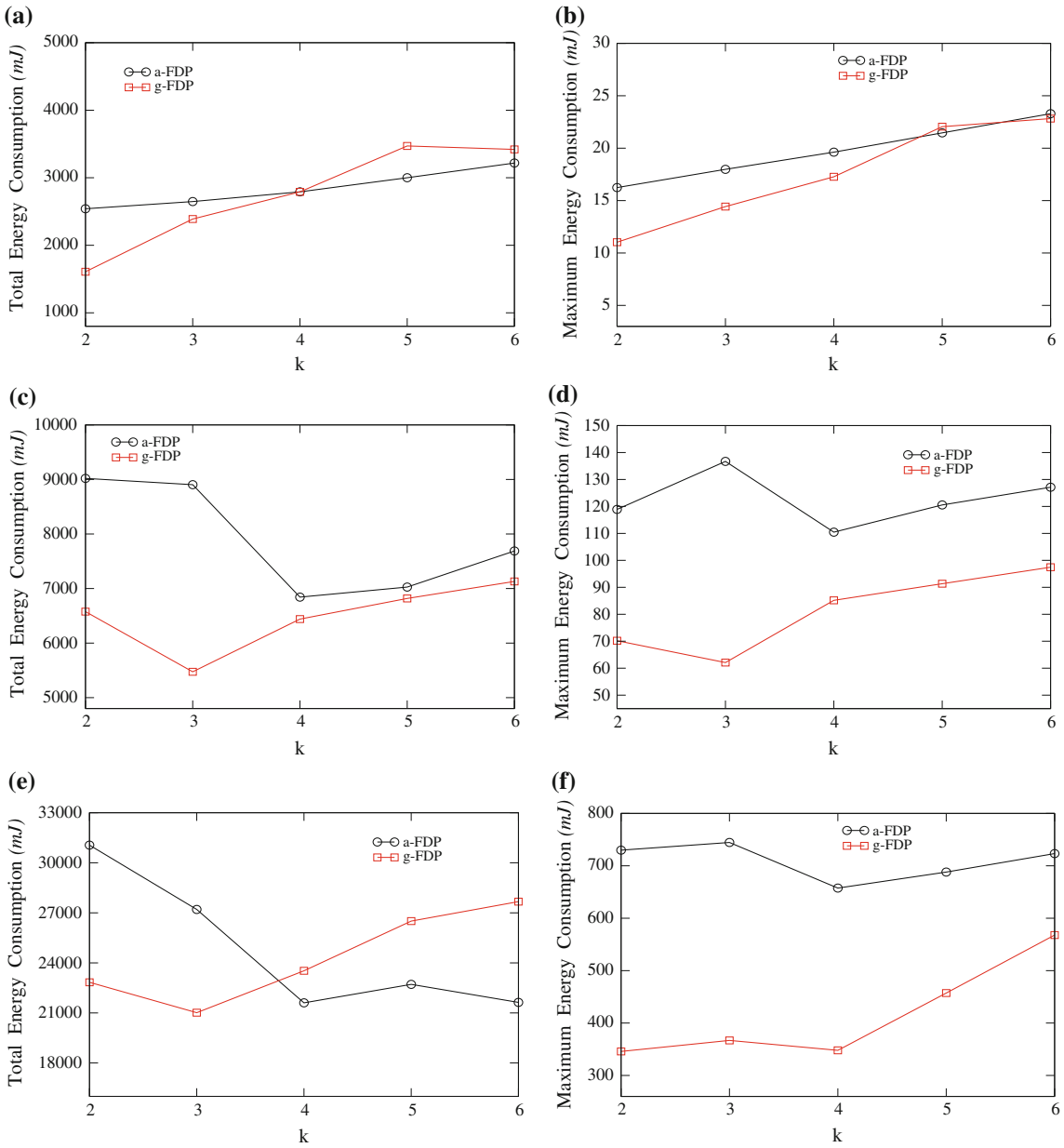


Fig. 6 The performance of algorithms a-FDP and g-FDP on independent datasets. **a** The total energy consumption with different values of k; **b** the maximum energy consumption with different values of k; **c** the total energy consumption with different values of k; **d** the maximum energy consumption with different values of k; **e** the total energy consumption with different values of k; **f** the maximum energy consumption with different values of k

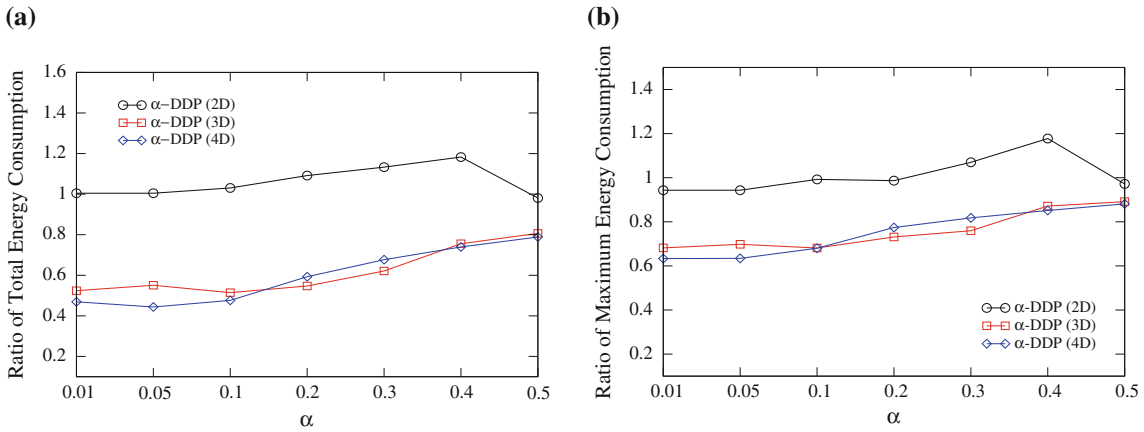


Fig. 7 The performance of algorithm α-DDP on independent datasets. **a** The total energy consumption with different values of α; **b** the maximum energy consumption with different values of α

6.3 The choice of α in algorithm α-DDP

Next, we evaluate the influence of different values of α on the performance of algorithm α-DDP. Figure 7(a), (b) plot the curves of ratio between the total energy consumption and the maximum energy consumption among the sensors by algorithm α-DDP to those by algorithm 1-DDP ($\alpha = 1$). It can be seen that algorithm α-DDP exhibits better performance than algorithm 1-DDP in both metrics, implying that setting $\alpha < 1$ leads to the energy savings from transmission. The curves on different datasets increase gradually as α increases, which means that the impact of α is minor and does not compromise the performance. It is concluded that an appropriate α should be chosen for different datasets.

6.4 Performance analysis of evaluation algorithms

We then investigate the performance of the proposed algorithms g-FDP with $k = 3$ and α-DDP with $\alpha = 0.05$ against existing algorithms by varying the dimensionality d from 2 to 4. We refer to the dynamic filter algorithm in [12] as algorithm DF, the single point filter algorithm and the grid index filter algorithm in [30] as algorithms TF and GI, the certificate filter algorithm in [17] as algorithm Cerf, respectively.

Figure 8 illustrates the ratios of the total energy consumption and the maximum energy consumption among the sensors by various algorithms on 2, 3, and 4 dimensional datasets to those by the skyline_merge algorithm. Figure 8(a), (b) show the performance of algorithms on correlated databases. It can be seen that overall algorithm α-DDP outperforms the existing algorithms. Algorithm GI consumes the most energy for query evaluation because the grid filter construction is costly compared to the energy consumption on single skyline query evaluation. Algorithm GI is more applicable to skyline maintenance, i.e., a grid filter is constructed and used for continuous queries. Figure 8(c), (d) indicate the performance of various algorithms on independent datasets, in which it can be seen that the total energy consumption and the maximum energy consumption among the sensors by algorithm α-DDP are the smallest on the independent datasets. Algorithm g-FDP performs better than the other algorithms except algorithm α-DDP on 3 and 4 dimensional datasets. Figure 8(e), (f) plot the ratios of performance by various algorithms on 2, 3, and 4 anti-correlated datasets. It can be seen that the performance on the anti-correlated datasets is worse than that on the independent datasets. On anti-correlated datasets, the number of skyline points in anti-correlated datasets is much more than that in independent datasets. Thus, fewer points will be filtered out, and consequently the energy savings by filtering unlikely skyline points from transmission becomes small. The total energy consumption by algorithm g-FDP is almost the same as that by algorithm Cerf, while the maximum energy consumption by algorithm g-FDP is smaller than that by algorithm Cerf. Among the mentioned algorithms, algorithm α-DDP performs the best. Figure 8(g), (h) illustrate the performance of various algorithms in the real datasets. It can be seen that algorithm g-FDP has a larger total energy consumption than that of algorithm Cerf on 2-dimensional datasets. But it performs better than algorithm Cerf on 3 and 4 dimensional datasets in terms of both metrics. Algorithm α-DDP outperforms the other algorithms remarkably on 2–4 dimensional datasets in both metrics. In conclusion, algorithm g-FDP is efficient in high dimensional datasets, and algorithm α-DDP is energy-efficient in various datasets and prolongs the network lifetime significantly.

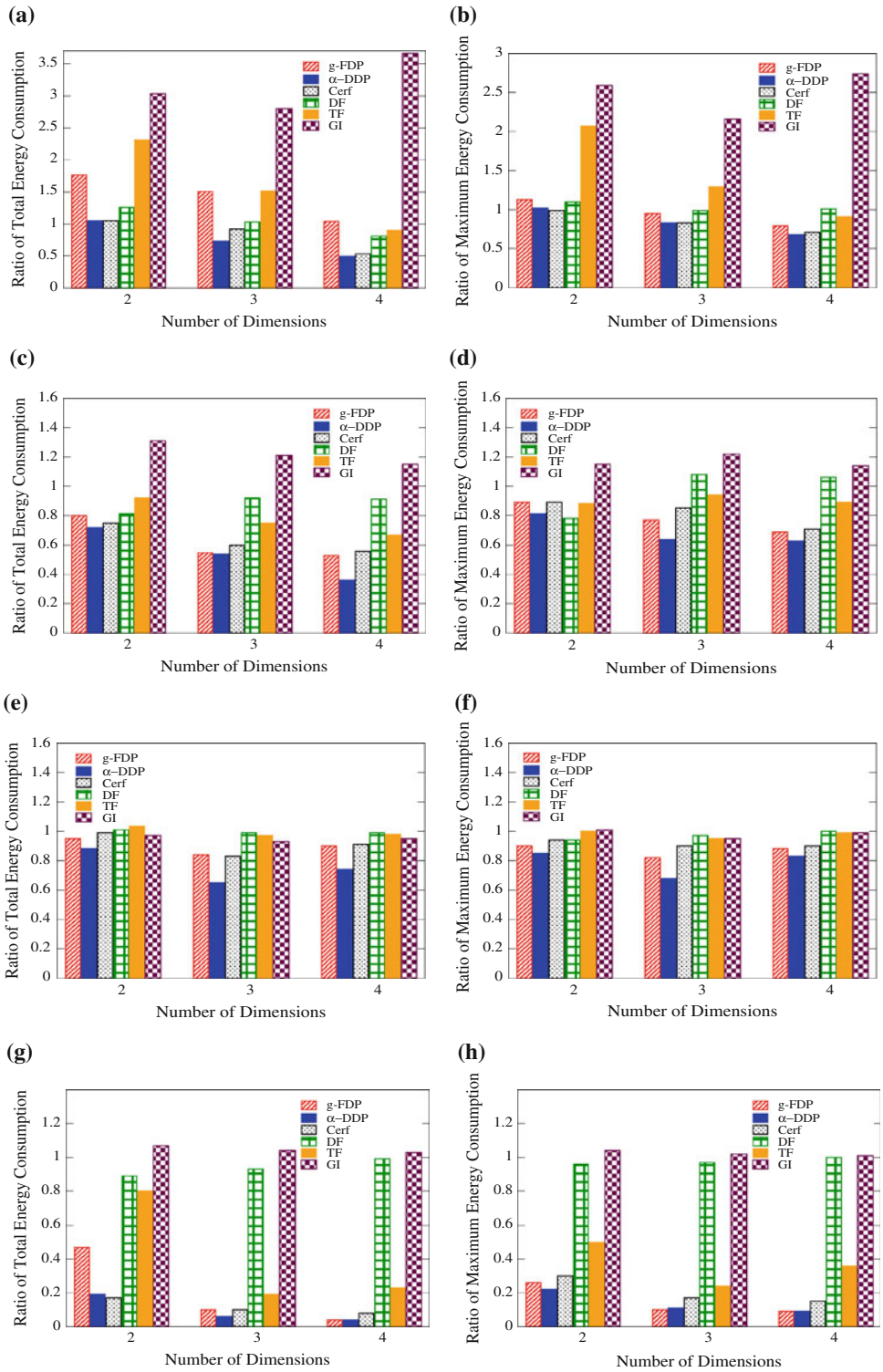


Fig. 8 The performance of evaluation algorithms on synthetic and real datasets. (**a**) Ratio of the total energy consumption on correlated datasets; (**b**) ratio of the maximum energy consumption on correlated datasets; (**c**) ratio of the total energy consumption on independent datasets; (**d**) ratio of the maximum energy consumption on independent datasets; (**e**) ratio of the total energy consumption on anti-correlated datasets; (**f**) ratio of the maximum energy consumption on anti-correlated datasets; (**g**) ratio of the maximum energy consumption on real datasets; (**h**) ratio of the maximum energy consumption on real datasets

6.5 Performance analysis of maintenance algorithms

Finally, we assess the performance of different algorithms for skyline maintenance in real datasets [11]. The window length is set to 300 time steps. Within each time step, only a fraction of sensors in the network generate new points. Specifically, the real sensed dataset is a collection of sensing data generated by a 54-sensor network [11]. To simulate the 54-sensor network into a 500-sensor network in our experiments, we use 10 sensors in our network to simulate the behaviors of one sensor in the 54-sensor network. If sensor v with ID i in the original network generates

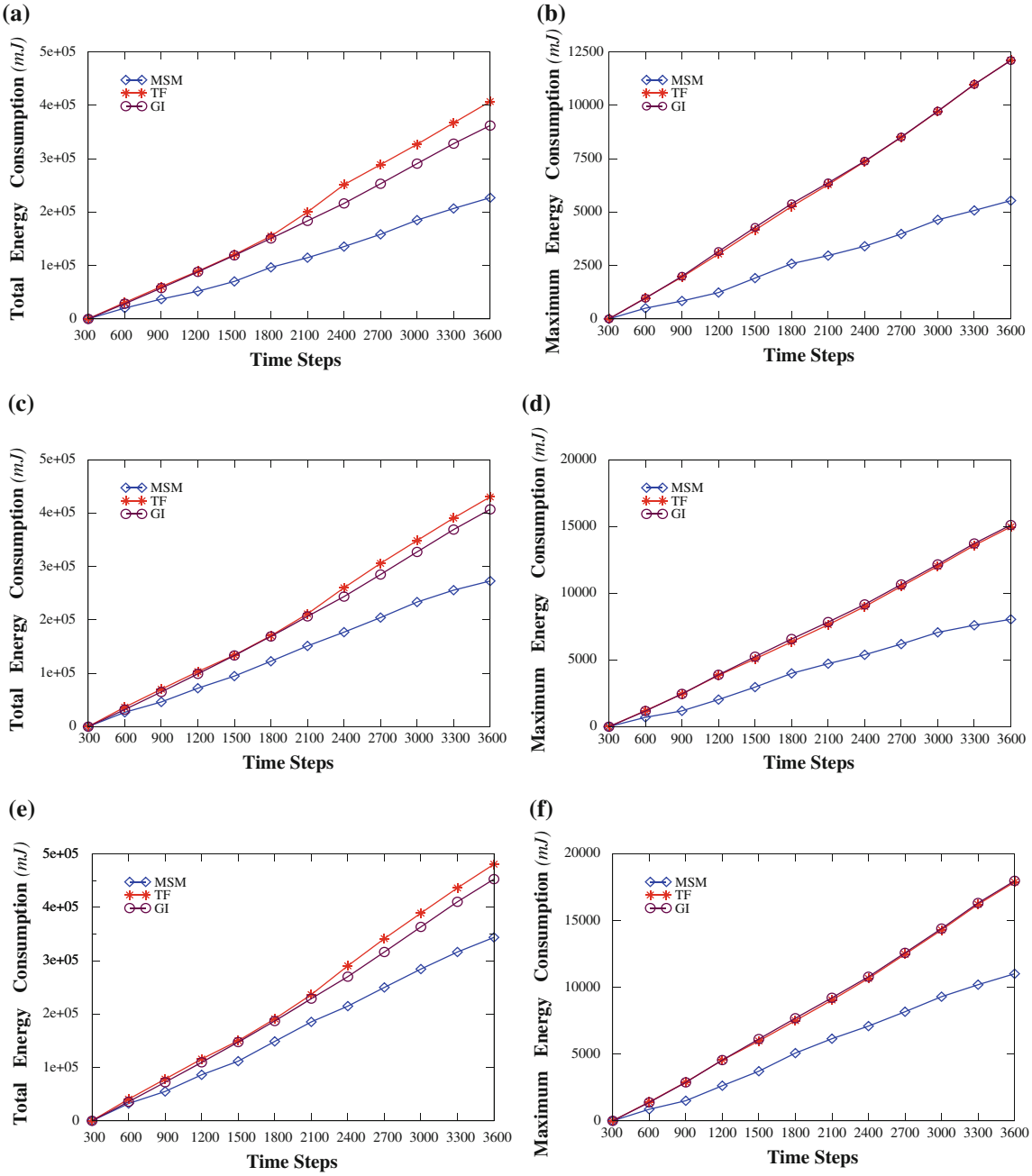


Fig. 9 The performance of various maintenance algorithms on real datasets. **a** Total energy consumption on 2 dimensional datasets; **b** maximum energy consumption on 2 dimensional datasets; **c** total energy consumption on 3 dimensional datasets; **d** maximum energy consumption on 3 dimensional datasets; **e** total energy consumption on 4 dimensional datasets; **f** maximum energy consumption on 4 dimensional datasets

a point at time step t, the corresponding sensor v' with ID $i' = (i*10 + t \ mod \ 10)$ in the 500-sensor network also generates the point at time step t, $1 \leq i \leq 50$.

Figure 9 plots the performance of different maintenance algorithms within sliding windows from time step 301 to 3,600 on the real datasets. It can be seen that algorithm `MSM` is the best among all algorithms in terms of the total energy consumption and the maximum energy consumption among the sensors. Algorithm `GI` outperforms algorithm `TF` in skyline maintenance due to ignoring the energy overhead on the construction of the grid filter, while the grid filter has better filtering capability than the single point filter. As time passes, the performance gap between algorithm `MSM` and algorithm `GI` becomes bigger and bigger, which implies that on average the energy consumption per query by algorithm `MSM` is smaller than that by algorithms `TF` and `GI`, and consequently the accumulated energy saving is significant. Thus, algorithm `MSM`

delivers a much longer network lifetime than that of existing algorithms.

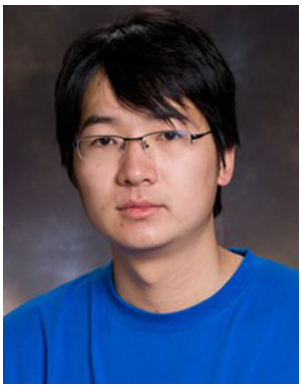
7 Conclusions

In this paper, we have studied skyline query evaluation and maintenance in energy-constrained wireless sensor network. We first devised three algorithms a/g-FDP and α-DDP for skyline query evaluation that progressively return the skyline points, by partitioning the entire dataset into several disjoint subsets and using the found skyline points to filter out unlikely skyline points from transmission in the network. We then proposed an energy-efficient incremental algorithm MSM for skyline maintenance within sliding window environments, by addressing which found skyline points to be selected for the global filter and at which time point the chosen skyline points should be broadcast. We finally conducted extensive experiments by simulation to evaluate the performance of the proposed algorithms against existing algorithms. The experimental results show that the proposed algorithms significantly outperform existing algorithms on both synthetic and real datasets in terms of various performance metrics.

References

1. Babcock, B., & Olston, C. (2003). Distributed top-*k* monitoring. In *Proceedings of SIGMOD*, pp. 28–39.
2. Bakle, W. T., Güntzer, U., & Zheng, J. X. (2004). Efficient distributed skylining for web information systems. In *Proceedings of EDBT*, pp. 256–273.
3. Börzsönyi, S., Kossmann, D., & Stocker, K. (2001). The skyline operator. In *Proceedings of ICDE*, pp. 421–430.
4. Chan, C. Y., Eng, P. K., & Tan, K. L. (2005). Stratified computation of skylines with partial-ordered domains. In *Proceedings of SIGMOD*, pp. 203–214.
5. Chen, H., Zhou, S., & Guan, J. (2007). Towards energy-efficient skyline monitoring in wireless sensor networks. In *Proceedings of European workshop on wireless sensor networks*. Lecture Notes in Computer Science, Vol. 4373, pp. 101–116.
6. Chen, L., Cui, B., Lu, H., Xu, L., & Xu, Q. (2008). iSky: Efficient and progressive skyline computing in a structured P2P network. In *Proceedings of ICDCS*, pp. 160–167.
7. Chen, B., Liang, W., & Yu, J. X. (2009). Progressive skyline query evaluation and maintenance in wireless sensor networks. In *Proceedings of CIKM*, pp. 1445–1448.
8. Chen, B., & Liang, W. (2009). Progressive skyline query processing in wireless sensor networks. In *Proceedings of MSN*, pp. 17–24.
9. Chomicki, J., Godfrey, P., Gryz, J., & Liang, D. (2003). Skyline with presorting. In *Proceedings of ICDE*, pp. 717–719.
10. Crossbow Inc. *MPR-Mote processor radio board users manual*.
11. (2004). http://db.csail.mit.edu/labdata/labdata.html.
12. Huang, Z., Jansen, C. S., Lu, H., & Ooi, B. C. (2006). Skyline queries against mobile lightweight devices in MANETs. In *Proceedings of ICDE*, pp. 66–76.
13. Kossmann, D., Ramask, F., & Rost, S. (2002). Shooting stars in the sky: An online algorithm for skyline queries. In *Proceedings of VLDB*, pp. 275–286.
14. Kwon, Y., Choi, J. H., Chung, Y. D., & Lee, S. K. (2007). In-network processing for skyline queries in sensor networks. *IEICE Transactions on Communication, E90-B*(12), 3452–3459.
15. Lee, K. C., Zheng, B., Lu, H., & Lee, W.-C. (2007). Approaching the skyline in Z order. In *Proceedings of VLDB*, pp. 279–290.
16. Li, C., Tung, A. K. H., Jin, W., & Ester, M. (2007). On dominating your neighborhood profitably. In *Proceedings of VLDB*, pp. 818–829.
17. Liang, W., Chen, B., & Yu, J. X. (2008). Energy-efficient skyline query processing and maintenance in sensor networks. In *Proceedings of CIKM*, pp. 1471–1472.
18. Lin, X., Yuan, Y., Wang, W., & Lu, H. (2005). Stabbing the sky:efficient skyline computation over sliding windows. In *Proceedings of ICDE*, pp. 502–513.
19. Lin, X., Yuan, Y., Zhang, Q., & Zhang, Y. (2007). Selecting stars: The k most representative skyline operator. In *Proceedings of ICDE*, pp. 86–95.
20. Madden, S., Franklin, M. J., Hellerstein, J. M., & Hong, W. (2002). TAG: A tiny aggregation service for ad hoc sensor networks. In *Proceedings of OSDI*, pp. 131–146.
21. Madden, S., Franklin, M. J., Hellerstein, J. M., & Hong, W. (2003). The design of an acquisitional query processor for sensor networks. In *Proceedings of SIGMOD*, pp. 491–502.
22. Papadias, D., Tao, Y., Fu, G., & Seeger, B. (2003). An optimal and progressive algorithm for skyline queries. In *Proceedings of SIGMOD*, pp. 467–478.
23. Pei, J., Jiang, B., Lin, X., & Yuan, Y. (2007). Probabilistic skylines on uncertain data. In *Proceedings of VLDB*, pp. 15–26.
24. Pottie, G. J., & Kaiser, W. J. (2000). Wireless integrated network sensors. *Communication of the ACM, 43*(5), 51–58.
25. Tan, K. L., Eng, P. K., & Ooi, B. C. (2001). Efficient progressive skyline computation. In *Proceedings of VLDB*, pp. 301–310.
26. The Network Simulator-ns2. (2006). http://www.isi.edu/nsnam/ns.
27. Wang, S., Vu, Q., Ooi, B. C., Tung, A. K. H., & Xu, L. (2009). Skyframe: A framework for skyline query processing in peer-to-peer systems. *The VLDB Journal, 18*, 345–362.
28. Wu, P., Zhang, C., Feng, Y., Zhao, B. Y., Agrawal, D., & Abbadi, A. E. (2006). Parallelizing skyline queries for scalable distribution. In *Proceedings of EDBT*, pp. 112–130.
29. Wu, M., Xu, J., Tang, X., & Lee, W.-C. (2007). Top-*k* monitoring in wireless sensor networks. *IEEE Transactiond on Knowledge and Data Engineering, 19*(7), 962–976.
30. Xin, J., Wang, G., Chen, L., Zhang, X., & Wang, Z. (2007). Continuously maintaining sliding window skyline in a sensor network. In *Proceedings of DASFAA*. Lecture Notes in Computer Science, Vol. 4443, pp. 509–521.
31. Yang, X., Lim, H. B., Ozsu, M. T., & Tan, K.-L. (2007). In-network execution of monitoring queries in sensor networks. In *Proceedings of SIGMOD*, pp. 521–532.
32. Yao, Y., & Gehrke, J. (2002). The cougar approach to in-network query processing in sensor networks. *ACM SIGMOD Record, 31*, 9–18.

Author Biographies



Baichen Chen received the Ph.D. degree from the Australian National University in 2012, the M.E. and the B.Sc. degree from Northeastern University, China in 2007 and 2004 respectively, all in computer science. He is currently a financial software engineer in Research and Development Department of Bloomberg company, UK. His research interests include information processing in wireless sensor networks, design and analysis of distributed algorithms and graph theory.

Weifa Liang received the Ph.D. degree from the Australian National University in 1998, the M.E. degree from the University of Science and Technology of China in 1989, and the B.Sc. degree from Wuhan University, China in 1984, all in computer science. He is currently an Associate Professor in the Research School of Computer Science at the Australian National University. His research interests include design and analysis of energy-efficient routing protocols for wireless ad hoc and sensor networks, information processing in wireless sensor networks, cloud computing, design and analysis of parallel and distributed algorithms, combinatorial optimization, and graph theory. He is a senior member of the IEEE.

Jeffrey Xu Yu received the B.E., M.E., and Ph.D. degrees in computer science, from the University of Tsukuba, Japan, in 1985, 1987, and 1990, respectively. Currently he is a Professor in the Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong. His major research interests include graph mining, graph database, keyword search, and query processing and optimization. He is a senior member of the IEEE, a member of the IEEE Computer Society, and a member of ACM.