Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 149513, 11 pages
http://dx.doi.org/10.1155/2015/149513

Research Article

Energy-Efficient β-Approximate Skylines Processing in Wireless Sensor Networks

Junchang Xin,[1,2] Zhiqiong Wang,[3] Mei Bai,[1,2] Linlin Ding,[4] and Guoren Wang[1,2]

[1]*College of Information Science & Engineering, Northeastern University, Shenyang 110819, China*
[2]*Liaoning Key Lab of Big Data Management & Analysis, Northeastern University, Shenyang 110819, China*
[3]*Sino-Dutch Biomedical & Information Engineering School, Northeastern University, Shenyang 110819, China*
[4]*College of Information Science & Technology, Liaoning University, Shenyang 110036, China*

Correspondence should be addressed to Junchang Xin; xinjunchang@ise.neu.edu.cn

Received 29 November 2014; Accepted 16 February 2015

Academic Editor: Vladimir Turetsky

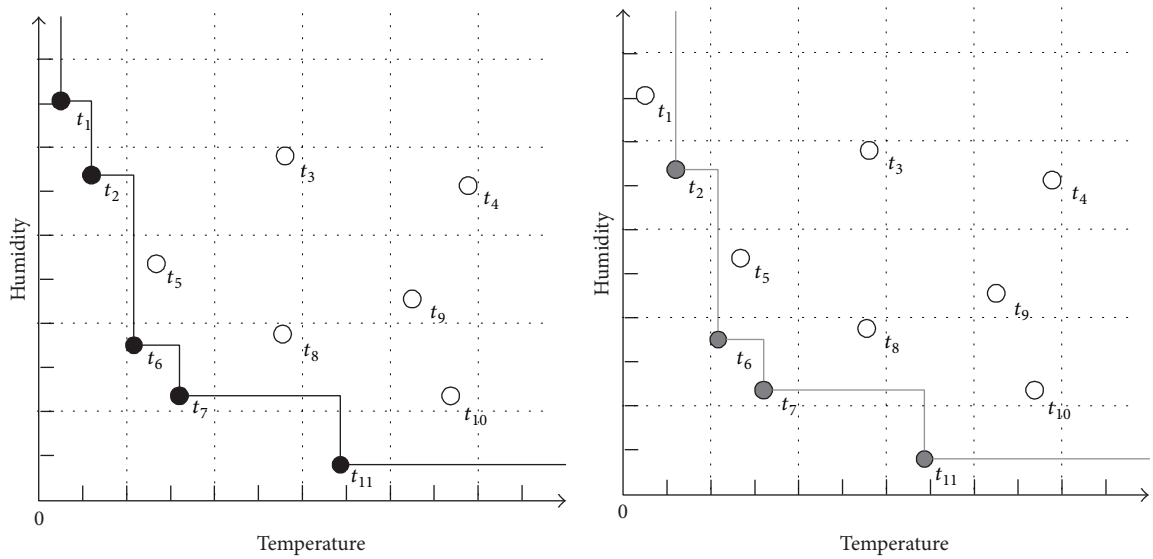
As the first priority of query processing in wireless sensor networks is to save the limited energy of sensor nodes and in many sensing applications a part of skyline result is enough for the user's requirement, calculating the exact skyline is not energy-efficient relatively. Therefore, a new approximate skyline query, β-approximate skyline query which is limited by a guaranteed error bound, is proposed in this paper. With an objective to reduce the communication cost in evaluating β-approximate skyline queries, we also propose an energy-efficient processing algorithm using mapping and filtering strategies, named Actual Approximate Skyline (AAS). And more than that, an extended algorithm named Hypothetical Approximate Skyline (HAS) which replaces the real tuples with the hypothetical ones is proposed to further reduce the communication cost. Extensive experiments on synthetic data have demonstrated the efficiency and effectiveness of our proposed approaches with various experimental settings.

1. Introduction

Wireless sensor networks (WSNs), which integrate sensor technology, embedded computing, networks, wireless communication, and distributed information processing, are widely used in military and civil fields [1, 2], such as object tracking, nuclear reactor controlling, fire detection, and battlefield surveillance. A WSN is a wireless network consisting of spatially distributed autonomous sensor nodes which are densely deployed either inside or close to the phenomenon and cooperatively monitor physical or environmental conditions, such as temperature, sound, vibration, humidity, and pressure at different locations. Sensor nodes are generally cheap, resource-constraint, unreliable, and battery powered; moreover most WSNs work in an unattended, hard-to-reach environment; it is impossible or at least very difficult to change batteries. Therefore, applications over WSNs need a scalable, energy-efficient, and fault-tolerant method to manage tremendous data generated by sensors and minimize power consumption to prolong the lifetime of WSNs.

As an important operator for multicriteria decision making and data mining, skyline query [3] has been well studied in the database literature. Given a dataset D, the skyline Skyline(D) of D returns all tuples that are not dominated by the others in D. Here, tuple t_i dominates another one t_j means t_i is no worse than t_j for every dimension and better than t_j for at least one dimension. The vocabularies "worse" and "better" mentioned above can be any preference judgment. Without loss of generality, we assume the smaller values are preferred in this paper. As illustrated in Figure 1(a), t_1, t_2, t_6, t_7, and t_{11} are the skyline tuples.

Actually, in many sensing applications, we just need a part of skyline result rather than the whole one, because that is enough for users to make decisions. For example, administrator always concerns the nodes with heavier traffic and lower battery to find out how the whole WSN works, so the administrator can take some actions to prolong the lifespan of WSNs. In this application, administrator just needs a rough condition about the relationship between traffics and power consumption, while evaluating skyline query in WSNs is



(a) Skyline (b) Approximate skyline

Figure 1: Examples of skyline and approximate skyline.

costly in terms of energy consumption, so it is not necessary to compute the exact skyline and an approximate one which can show the general shape of skyline is enough.

So far, approximate skyline has already been studied in traditional database literature. However, the existing solutions cannot be applied to the sensor environment directly due to the unique characteristics of WSN. In WSNs, energy is the precious resource and wireless communication is the main consumer [4], and the main challenge of approximate skyline processing in WSNs is how to minimize the communication cost. Although an approximate skyline which returns a subset of the results over WSNs was studied in [5], its definition may be not good enough to represent the exact skyline, as it did not guarantee the error bound. Therefore, in this paper, a new approximate skyline definition is proposed which can return the subset of a skyline in certain error bound. Moreover, the corresponding energy-efficient query processing algorithms are proposed accordingly. The contributions of this paper are summarized as follows.

(i) An error-bounded β-approximate skyline is proposed, and an Actual Approximate Skyline (AAS) algorithm based on mapping and filtering to solve β-approximate skyline is also proposed to reduce the communication cost among sensor nodes on evaluating β-approximate skyline in WSNs.

(ii) Hypothetical Approximate Skyline (HAS) algorithm, which not only is limited by error bound but also replaces several real tuples with the hypothetic ones, is also proposed to further improve the query processing efficiency.

(iii) Last but not least, our extensive experimental studies using synthetic data show that the proposed approaches can significantly reduce the communication cost among sensor nodes and save the energy consumption during the evaluation processing of approximate skyline queries in WSNs.

The rest of the paper is organized as follows. Section 2 briefly reviews the previous related work. The proposed Actual Approximate Skyline (AAS) and Hypothetical Approximate Skyline (HAS) algorithms are introduced in Sections 3 and 4, respectively. The extensive experimental evaluation results showing the effectiveness and energy-efficiency of the proposed approximate skyline algorithms are reported in the Section 5. Finally, Section 6 concludes this paper.

2. Related Work

The skyline operator was first introduced in [3], where two algorithms based on block nested loops (BNL) and divide-and-conquer (D&C) were proposed, respectively. As a variant of BNL, a sort-filter-skyline (SFS) algorithm which improves the performance by presorting the dataset according to some monotone scoring function was proposed in [6]. Two progressive processing algorithms, Bitmap and Index, were proposed in [7]. Both of them can obtain the skyline result set without having scanned the whole dataset. A nearest neighbor (NN) approach was investigated in [8], which can process skyline query progressively by using the result of the nearest neighbor query to partition the data space recursively. Papadias et al. [9] proposed an algorithm based on branch-and-bound (BBS) taking advantage of R-tree to improve the performance of NN. Besides the original skyline definition, there are a lot of skyline variants having been proposed. Jin et al. [10] suggested the conception of thick skyline offering more results for users to choose. An approximate skyline algorithm based on BBS was proposed in [11]. Chan et al. [12] relaxed the idea of dominance to k-dominance. A novel

metric, called skyline frequency that compares and ranks the interestingness of data points, was considered in [13]. Lin et al. [14] studied the problem of selecting k skyline points so that the number of points, which are dominated by at least one of these k skyline points, is maximized. Benouaret et al. [15, 16] introduce two new concepts based on an extension of the (Pareto) dominance relationship, called α-dominant skyline [15] and σ-dominant skyline [16], to tackle multicriteria service selection, and then proposed an efficient and flexible Web service selection framework that implement the above mentioned skyline variants [16]. All these methods introduced above are just available in the centralized database system. In the distributed database system especially in WSNs, the situation is more complex. In WSNs, not only data is storage distributed but also the nodes are battery supported. As a result, energy-efficiency should be the first priority of the algorithm designed in WSNs considering the limited battery power in the sensor nodes.

Various skyline processing algorithms in WSNs have been studied recently. Chen et al. [17] presented a hierarchical threshold-based approach to minimize the transmission traffic in WSNs. Xin et al. [18] presented a filter-based approach which employs two types of filters (tuple filter and grid filter) within each sensor to reduce the cost of transmission traffic in WSNs. While in [19] they also presented an energy-efficient approach which uses the mapped skyline as the filter to get to the purpose of energy-efficiency in WSNs. In [20], The filter-based distributed algorithms for skyline evaluation and maintenance were studied to maximize the network lifetime. In [21], the dataset is partitioned into several disjoint subsets and skyline points can be found progressively by examining each subsequent subset. Multiple skyline queries over WSNs were discussed in [22], and an energy-efficient multi-skyline evaluation (EMSE) algorithm which can reduce the transmission cost with two optimization mechanism was investigated here. Shen et al. [23] studied two-dimensional skyline query based on position in WSNs and proposed Ring-Skyline algorithm which calculates the skyline for each ring according to the order of distance from the near to the far. Su et al. [24] proposed a data-centric algorithm named Skysensor using a cluster-based architecture, and Skysensor can reduce the energy consumption for each query depending on several skyline queries started by different sensors sharing the same data gathering process. Pan et al. [5] investigated an approximate skyline (AS) algorithm which computes an approximate skyline result set only by making partial sensor nodes transmitting their sensor data back, and the approximate skyline result set is just a subset of the exact skyline set. This approach can reduce the energy cost efficiently, but it does not consider the error bound between the approximate skyline and exact skyline, which may cause the skyline result unilateral and inadequate. In this paper, we will propose a new definition of approximate skyline taking both inclusion relationship and error bound into consideration, and we call it β-approximate skyline. It always can show the general distribution of exact skyline that compensates for the weakness of approximate skyline in [5].

3. Approximate Skyline in Wireless Sensor Networks

In this section, the network routing structure and the definition of β-approximate skyline are first introduced in Section 3.1. Then, the preliminaries which are the foundations of our proposed approaches are presented in Section 3.2. Finally, the details of Actual Approximate Skyline (AAS) algorithm are described in Section 3.3. Notations section summarizes the notations used throughout the paper.

3.1. Problem Statement. The tree-based routing structure is established to tackle the β-approximate skyline in WSNs. It constructs a spanning tree with the base station as the root. The construction process is as follows: All nodes set their own level to infinite. The base station broadcasts a message with its own id and level to construct the routing tree. The level of base station is usually set to zero. Any node that hears the message will compare its own level to the level in the message; if the former is bigger, it will be replaced by the latter added by one and also chooses the sender as its parent. Each of these nodes then replaces the id and level with their own ids and levels and then rebroadcasts the routing message to their neighbors. The routing tree is constructed step by step this way. This construction process will be initiated periodically by the base station; thus the network topology will be constructed periodically. Therefore, this structure can easily adapt to the moving, addition, or deletion of the node.

Although the approximate skyline algorithm proposed by Pan et al. [5] can reduce communication cost efficiently in WSNs, it may seriously affect precision of the skyline result. Therefore, the skyline result cannot show general distribution of the exact skyline very well. In practical applications, the query result is "good" or "bad" which always depends on the precision of approximate skyline. Hence, besides considering that approximate skyline is the subset of the exact one, the distance between them should also be taken into consideration at the same time, to make sure approximate skyline can show the general distribution of exact skyline very well. Consequently, a new definition of approximate skyline named β-approximate skyline is proposed and shown as Definition 1.

Definition 1. Given a dataset D and its skyline being S, if there is a dataset $\widetilde{S}$ satisfying that $\widetilde{S}$ is a subset of S ($\widetilde{S} \subseteq S$) and the distance $\mathrm{Dis}(\widetilde{S}, S)$ between the surfaces of $\widetilde{S}$ and S satisfies $\mathrm{Dis}(\widetilde{S}, S) \leq \beta$, then it can be said that $\widetilde{S}$ is a β-approximate skyline of D.

Here, the surface is the shape of a region dominated by at least one of the elements in a set of skyline tuples. It is known that the surface is composed by the zipzag plane of the skylines and distance between two skyline surfaces is the maximum of all distances between a tuple in one skyline surface and the other surface (i.e., point to surface distance).

Figure 1(b) straightly illustrates the corresponding β-approximate skyline of the dataset in Figure 1(a). Comparing with the exact skyline in Figure 1(a), we can conclude that

```
for local β-approximate skyline S̃_i of each child node do
    D = D + S̃_i;
D' = Mapping(D, f);
for each tuple t in D do
    t' = getMapping(D', t);
    if t' is not dominated by any other tuples in D' then
        D̃ = ReverseMapping(D, t', f);
        if t is not dominated by any other tuples in D̃ then
            S̃ = S̃ + {t};
return;
```

Algorithm 1: Actual Approximate Skyline (AAS).

the dominating region of β-approximate skyline is almost the same with the exact skyline's. The distance between the exact skyline and the approximate one is no more than β.

3.2. Preliminaries. First, the definition of mapped skyline is introduced here.

Using a regular grid, the data space can be partitioned into many cells. For every dimension, assume that the extent of grid is β; then we will have $r[k] = \lceil (u[k] - l[k])/\beta \rceil$ segments. And totally there are $\prod_{k=1}^{d} r[k]$ cells in the data space. In other words, the division processing is equivalent to using mapping function $f(x) = \lfloor (x - l[k])/\beta \rfloor$ to map every value $x \in [l[k], u[k]]$ on dimension k for all the tuples to integer at a range of $[0, r[k] - 1]$. The mapped dataset D' can be gotten from original dataset D using the method above. The definition of mapped skyline is shown as Definition 2.

Definition 2. Given a dataset D, the skyline Skyline(D') of its mapped dataset D' is defined as mapped skyline of D.

Next, we will introduce how to calculate the β-approximate skyline depending on mapped skyline.

Lemma 3. *If $t_i \succeq t_j$, then $f(t_i) = f(t_j)$ or $f(t_i) \succeq f(t_j)$.*

Proof. If t_i and t_j belong to the same cell divided by β, then $f(t_i) = f(t_j)$. Otherwise, $f(t_i) \neq f(t_j)$, since t_i is no worse than t_j for all the dimensions and better than t_j for at least one dimension, according to the mapping function, we can easily get that $f(t_i)$ is no worse than $f(t_j)$ for all dimensions and better than $f(t_j)$ for at least one dimension; therefore, $f(t_i) \succeq f(t_j)$. The proof is completed. □

According to Lemma 3, we can get the important Theorem 4, which is the foundation of AAS algorithm.

Theorem 4. *Given a dataset D, $\widetilde{D}$ presents the tuples in D whose mapped ones belong to Skyline(D'); namely, $\widetilde{D} = \{t \in D, f(t) \in Skyline(D')\}$. Then Skyline($\widetilde{D}$) is β-approximate skyline of D.*

Proof. According to Definition 1, two aspects need to be proofed.

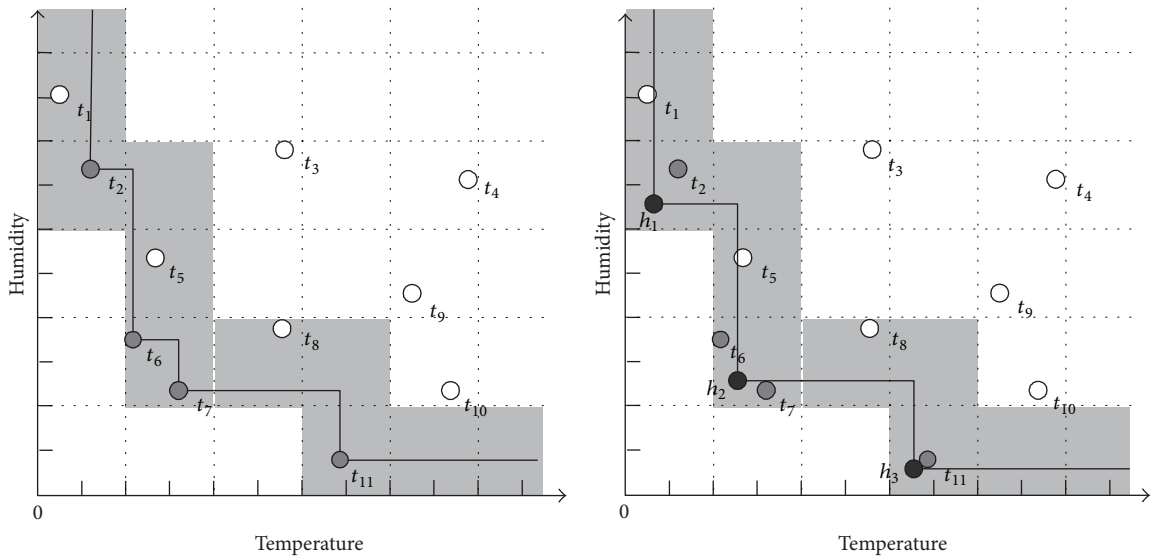
(1) Skyline($\widetilde{D}$) $\subseteq$ Skyline(D). Assume that $t_j \notin$ Skyline(D); there must be another tuple $t_i \in$ Skyline(D) which can dominate t_j. Based on Lemma 3, we are sure that $f(t_i) = f(t_j)$ or $f(t_i) \succeq (t_j)$. In the case of $f(t_i) = f(t_j)$, both t_i and t_j belong to $\widetilde{D}$, and $t_i \succeq t_j$, so $t_j \notin$ Skyline($\widetilde{D}$). In the other case of $f(t_i) \succeq f(t_j)$, we know that $t_j \notin \widetilde{D}$ that directly results in $t_j \notin$ Skyline($\widetilde{D}$).

(2) Dis(Skyline($\widetilde{D}$), Skyline(D)) $\leq \beta$. Depending on the mapping function, the distance between t and $f(t)$ for each dimension is no more than β; therefore the distance between Skyline(D) and Skyline(D') or rather $\widetilde{D}$ is no more than β. And in the calculation of Skyline($\widetilde{D}$), the distance between surfaces of two skyline dataset cannot exceed β if we only test the dominance relationship among tuples which have the same mapped one. So Dis(Skyline($\widetilde{D}$), Skyline(D)) $\leq \beta$.

In conclusion, Skyline($\widetilde{D}$) is the β-approximate skyline of dataset D. □

3.3. Actual Approximate Skyline Algorithm. According to Theorem 4, the calculation processing of β-approximate skyline of Dataset D, namely, ApprSkyline$_\beta(D)$ consists of three steps. First, calculate which tuples in D' belong to Skyline(D'). Then, all the tuples whose mapped one $f(t)$ is in Skyline(D') should be figured out and kept in $\widetilde{D}$. Finally, we evaluate Skyline($\widetilde{D}$) as β-approximate skyline of D.

The in-network computation [4] is used to calculate the β-approximate skyline in WSNs, that is to compute local query results of in-network whenever possible. Firstly, the leaf sensor nodes in the routing tree calculate their local β-approximate skyline and send them to their parent nodes, respectively. Then, the intermediate sensor node merges its local sensing data and the β-approximate skyline results sent by its children firstly and then sends the merged intermediate result to its parent. Finally, the base station will get the global β-approximate skyline of WSNs. The query processing in each sensor node (including leaf nodes, intermediate nodes, and base station) is shown as Algorithm 1. First, in the *merging and mapping* step, we merge the β-approximate skyline $\widetilde{S}_i$ of each child node into the dataset D of this current node and then get the mapped dataset D' of D using the



(a) β-approximate skyline (b) β-hypothetical skyline

Figure 2: Illustration of approximate skylines.

mapping function. Second, in the *dominace judgement* step, we get each tuple $t \in D$ and its mapped tuple $t' \in D'$; if t' is the mapped skyline tuple, then we should figure out whether t is dominated by any of the real tuples in the same cell corresponding to t' according to the mapping function. If it is not, t is the skyline of this cell and t must belong to the β-approximate skyline and add t to local β-approximate skyline result $\widetilde{S}$. When all the sensor nodes as well as the base station have finished their query processing, we can get the final β-approximate skyline result $\widetilde{S}$.

4. Hypothetical Approximate Skyline

Since using AAS algorithm to calculate approximate skyline in WSNs can reduce the communication cost significantly comparing with calculating the exact one, it can efficiently extend the lifespan of WSNs. While in some sensing applications, users just need to know the distribution of skyline roughly rather than the real skyline tuple values. Consequently, we tend to maintain the error bound in β-approximate skyline but relax the restriction that β-approximate skyline should be the subset of exact skyline. To deal with that, we use the expected tuple to replace all the tuples in the cell which contains β-approximate skyline; then we can get algorithm β-hypothetical skyline named $\text{ApprSkyline}_{\beta}^{H}(D)$ based on the expected tuples which are also called hypothetical tuples in this paper. As Figure 2 shows, both β-hypothetical skyline and β-approximate skyline can present the general distribution of exact skyline very well, while the number of tuples is less in β-hypothetical skyline.

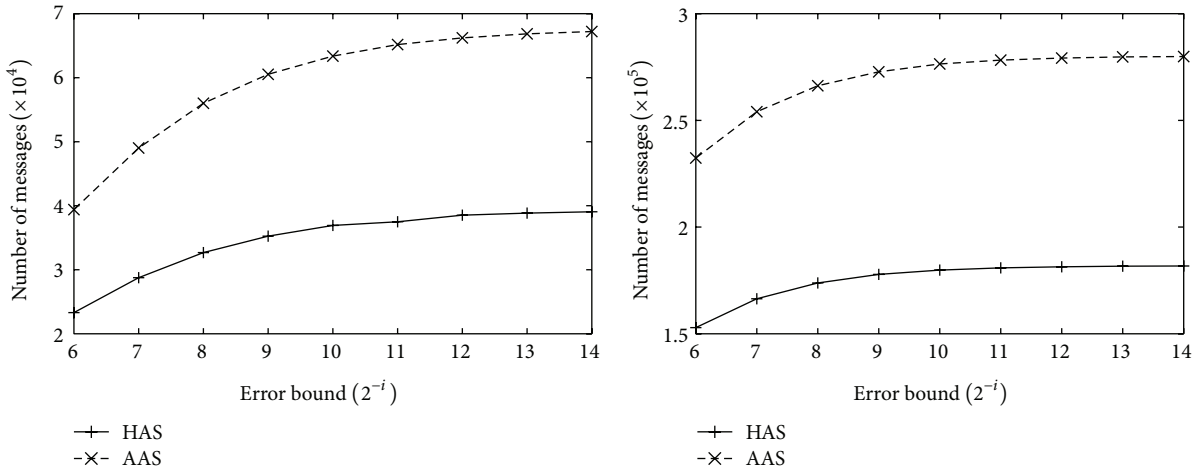
In the calculation processing of β-hypothetical skyline, we need to evaluate mapped skyline in the whole WSNs first. Then in the base station, we figure out the hypothetical tuples corresponding to all the mapped tuples in the mapped skyline, and these hypothetical tuples make up the β-hypothetical skyline result in WSNs. In this way, not only cannot the distance between hypothetic skyline and exact skyline, namely, $\text{Dis}(\text{ApprSkyline}_{\beta}^{H}(D), \text{Skyline}(D))$, exceed β, but also the quantity of data transferred in WSNs can be reduced significantly since the communication cost of mapped tuples is far more less than that of real tuples. The reasons of that are that the number of mapped tuples is less than the real ones and the values of mapped tuples are integer while the real tuples are real number type data. Actually, the advantage of integer type data transmission has already been demonstrated in [19].

To get the hypothetical tuples which also mean expected tuples, we can use the method mentioned in [11], which is shown as follows: taking advantage of the equation $E(\lambda) = 1/(d \times N + 1)$, the expected value of skyline tuple λ in an unit cell can be calculated; then we can get the expect tuple using the equation $\widetilde{x} = l + \lfloor (x - l)/\beta \rfloor \times \beta + \beta/(d \times N + 1)$.

The Hypothetical Approximate Skyline (HAS) algorithm, which can achieve β-hypothetical skyline result is shown as Algorithm 2. We should get the mapped dataset D' of this node first and then merge the local mapped skyline S_i' of each child node into D'. After all these have been done, we can get the mapped skyline S' of D'. Finally, the base station aggregates all the mapped skyline and figures out the ultimate mapped skyline and then calculates the β-hypothetical skyline result according to S' using the equations mentioned above.

5. Performance Evaluation

In this section, we will present our simulation results comparing the performance of our algorithms using independent, correlated, and anticorrelated data [3]. The result of correlated



(a) Independent (b) Anticorrelated

Figure 3: Number of message versus error bound.

$D' = Mapping(D, f)$;
For *local mapped skyline* S'_i *of each child node* **do**
$D' = D' + S'_i$;
$S' = Skyline(D')$;
return;

Algorithm 2: Hypothetical Approximate Skyline (HAS).

Table 1: Simulation parameters.

Parameter	Range and default
Error bound (β)	$0.5^6, 0.5^7, 0.5^8, 0.5^9, 0.5^{10}, 0.5^{11}$, $\mathbf{0.5^{12}}$, $0.5^{13}, 0.5^{14}$
Number of nodes (N)	600, 700, 800, 900, **1000**
Dimensionality (d)	2, **3**, 4, 5
Cardinality (c)	100, 200, **300**, 400, 500

data is omitted, since it is similar with the performance of independent data. In order to prove that approximate skyline algorithm is more energy-efficient than the exact skyline algorithm, we also take TAG into consideration. So, the algorithms we need to compare are

(i) TAG: a Tiny AGgregation service for ad hoc sensor networks [4];

(ii) AAS: Actual Approximate Skyline algorithm;

(iii) HAS: Hypothetical Approximate Skyline algorithm.

5.1. Experimental Settings. We have developed a simulator using java to evaluate the performance of our proposed algorithms, and the parameters of simulator are error bound, number of nodes, dimensionality, and cardinality. In our experiment, we place n sensor nodes in an area of $\sqrt{n} \times \sqrt{n}$ unit at random; then each node holds one unit space averagely and the communication radius of nodes is set to $2\sqrt{2}$ unit. Meanwhile, we make the capacity of packet transmitted in the network be no more than 48 bytes [6]. Table 1 presents the parameters we investigate along with their default values and ranges in our experiments. We vary one single parameter and keep the others being their default values in every experiment. All the simulations are run on the PC with 2.8 GHz CPU and 512 M of memory.

The performance metrics of our experiments include communication cost (number of messages) and result quality (relative error). Result quality is measured by relative error which is the ratio of area of relative difference between S and $\widetilde{S}$ to the area dominated by S. It can be denoted as

$$e_r = \frac{2D\left(S \cup \widetilde{S}\right) - D(S) - D\left(\widetilde{S}\right)}{D(S)} \times 100\%. \quad (1)$$

Particularly for β-approximate skyline, the area dominated by $\widetilde{S}$ is covered by that dominated by S. Thus, the formula can be simplified as

$$e_r = \frac{D(S) - D\left(\widetilde{S}\right)}{D(S)} \times 100\%. \quad (2)$$

5.2. Experimental Results. As shown in Figures 3(a) and 3(b), the less error bound becomes, the more communication cost will be since the number of mapped tuples will increase when we make error bound become smaller. However, the increasing rate of communication cost will begin to slow down when it reaches to a certain value, and this limit state can be regarded as a tendency towards TAG. In addition, we can find that the communication cost for HAS is always lower than that of AAS because transmitting the integer tuples is more energy-efficient than transmitting the tuples consisting of real numbers.

In Figures 4(a) and 4(b), we find that when error bound is minimal, relative error would reduce since number of

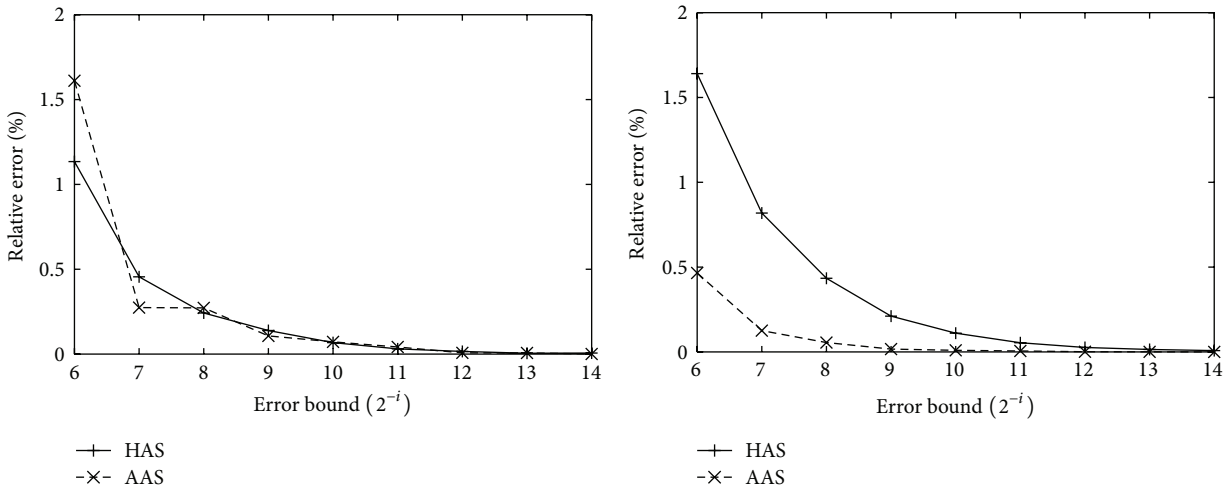


Figure 4: Relative error versus error bound.

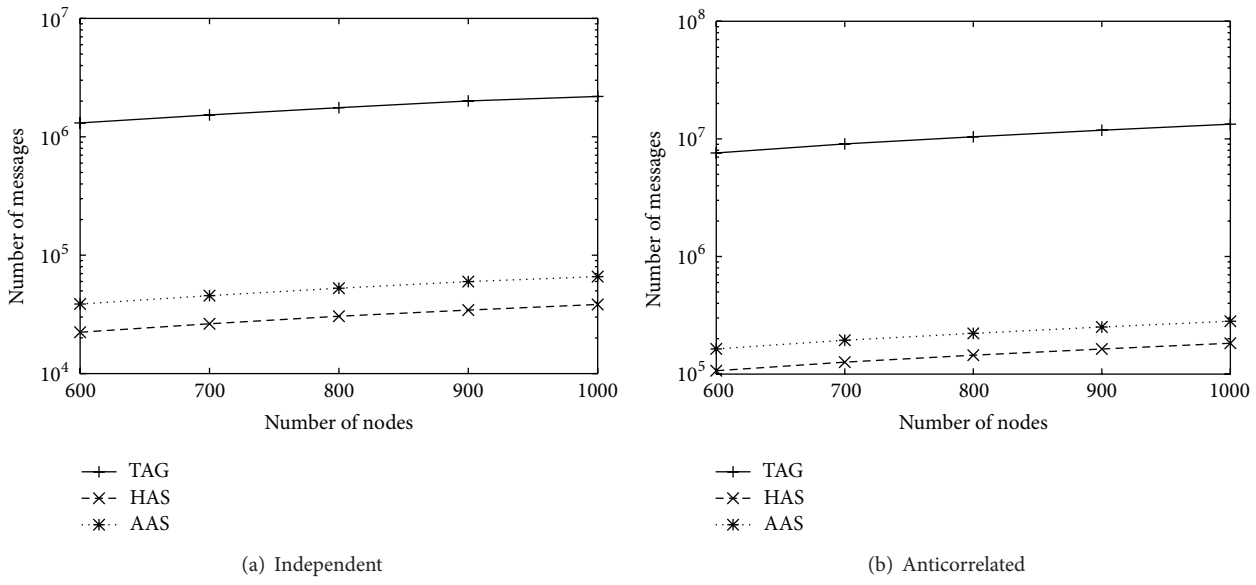


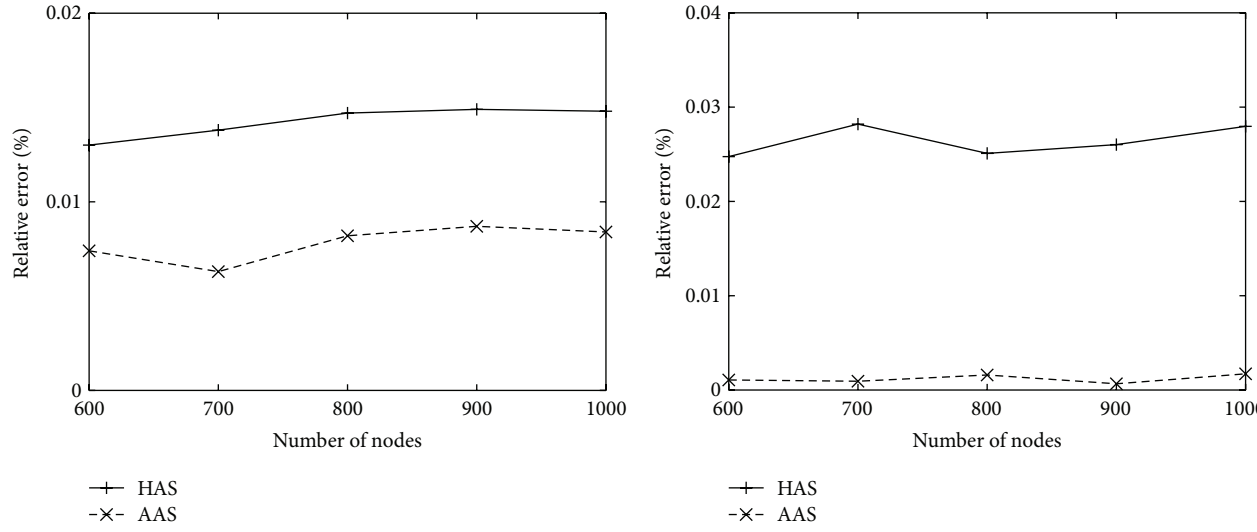
Figure 5: Number of message versus number of nodes.

skyline tuples calculated by our algorithms increases for the decreasing of error bound, which makes the skyline more precise. Moreover, relative error of HAS is always higher than that of AAS for the reason that result of AAS consists of the real skyline tuples but result of HAS is made up of hypothetical ones. Relative error is controlled well by setting error bound to some appropriate value, and when error bound reaches some critical value such as $1/2^{12}$, the relative error is near to zero. Its astringency is pretty good, and we make default error bound equal to $1/2^{12}$ in all experiments.

As shown in Figures 5(a) and 5(b), the communication cost increases correspondingly when the number of nodes increases. Communication cost for TAG is the highest and HAS has the least communication cost among the three algorithms. The reason is that TAG calculates the exact skyline result, while HAS transmits the integer tuples in the network and AAS uses tuples consisting of real numbers.

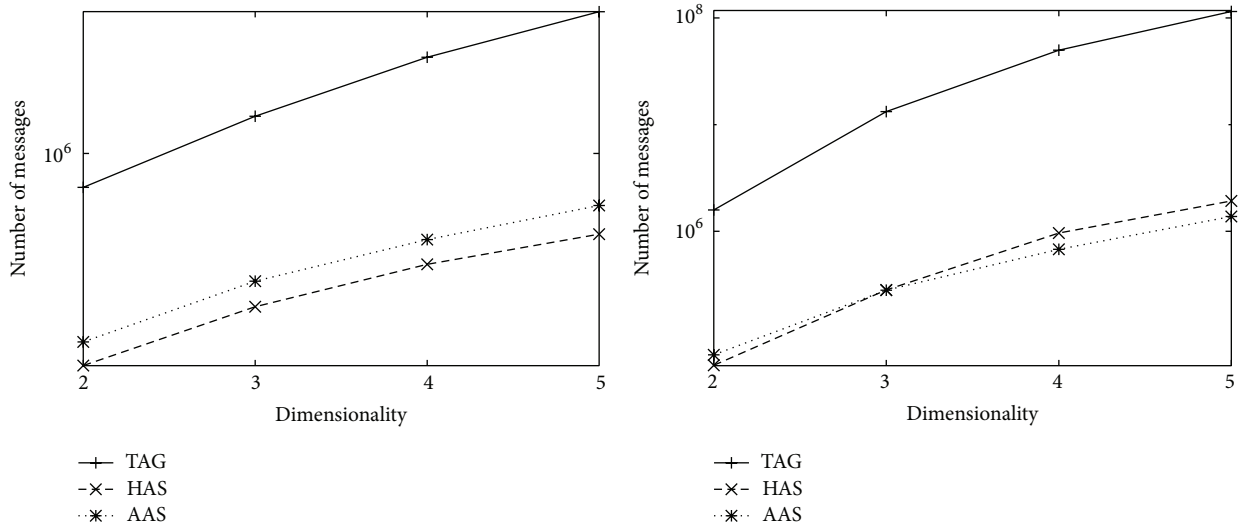
In Figures 6(a) and 6(b), since the distance between approximate skyline and exact skyline is controlled well with the mapping function, the change to number of sensors can just result in relative error's fluctuating, and the influence is not very significant. Moreover, the relative error of AAS fluctuates more slightly comparing with that of HAS.

Figures 7(a) and 7(b) show that the communication cost increases along with the dimensionality increase, because the rate of a tuple being dominated decreases with the increasing dimensionality, which gives rise to the amount of skyline tuples to be transmitted getting lager. And when dimensionality increases, relative error increases generally in Figures 8(a) and 8(b) since ratio of the region dominated by both



(a) Independent (b) Anticorrelated

Figure 6: Relative error versus number of nodes.



(a) Independent (b) Anticorrelated

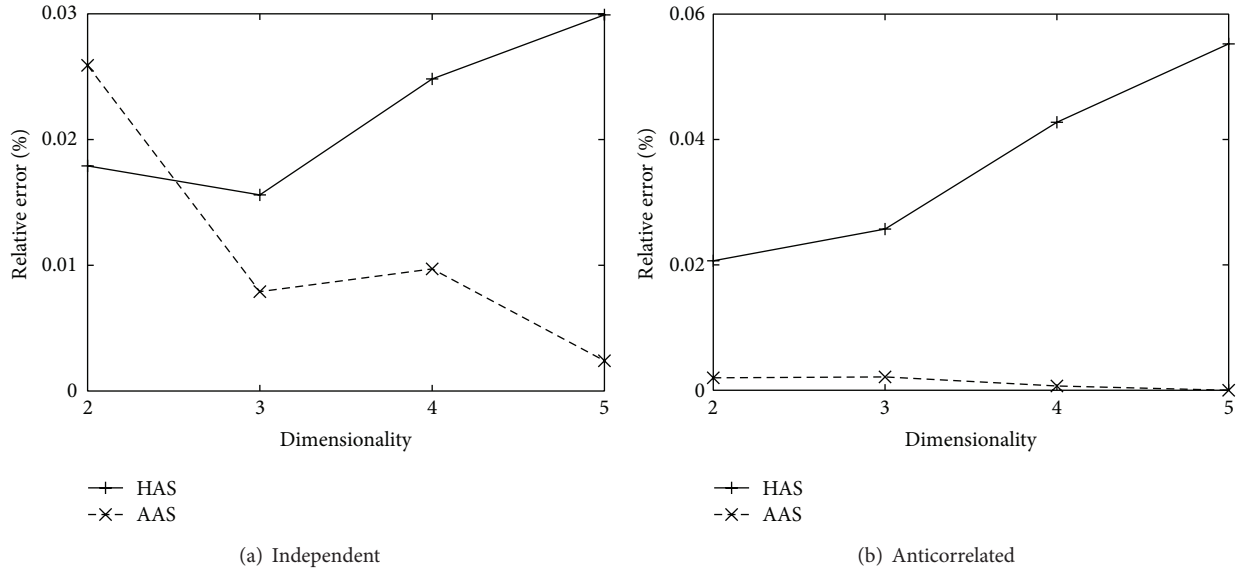
Figure 7: Number of message versus dimensionality.

approximate skyline and exact skyline becomes smaller with the increasing of dimensionality.

As shown in Figures 9(a) and 9(b), the communication cost for TAG is larger than that of the other two approximate algorithms. And the communication cost does not vary much with the changing of cardinality for all the three algorithms for the reason that error bound is a constant in this experiment. In Figures 10(a) and 10(b), we can see that when cardinality changes, both of the approximate algorithm's relative errors will change slightly. The reason is similar with what we present above if number of nodes varies.

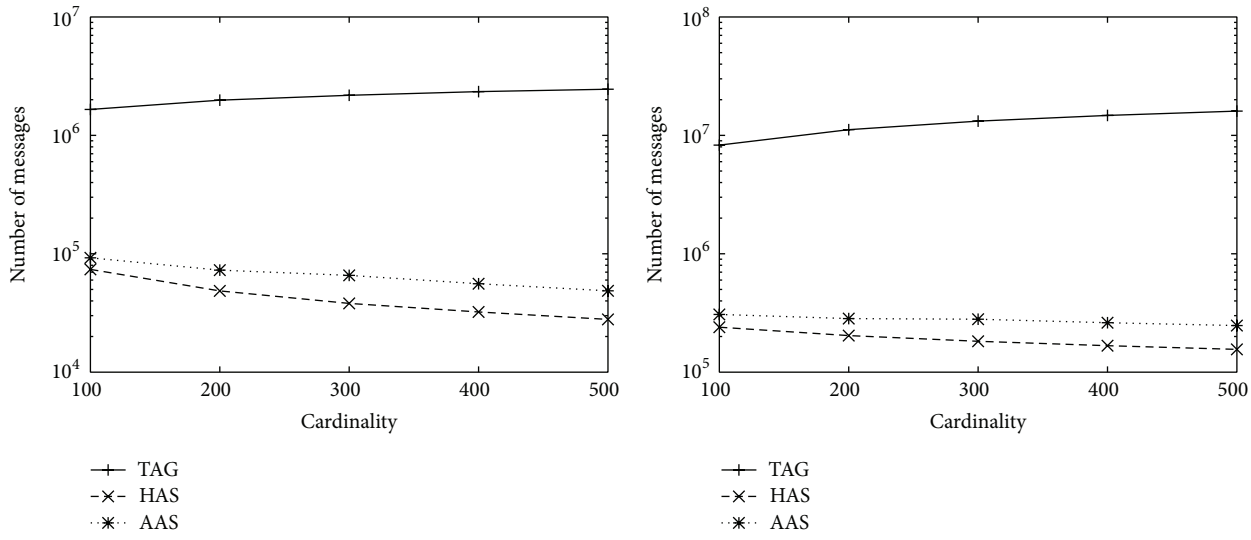
6. Conclusions

In most WSNs, energy is a critical resource and is mainly consumed by the wireless communication. How to minimize the communication cost in WSNs becomes an essential problem. In this paper, we presented a comprehensive study on approximate skyline queries in WSNs. First, we proposed the definition of β-approximate skyline, which demands that not only are the tuples in it the members of exact skyline but also the distance between the approximate skyline and the exact skyline is no more than β. Then, the AAS algorithm



(a) Independent (b) Anticorrelated

Figure 8: Relative error versus dimensionality.



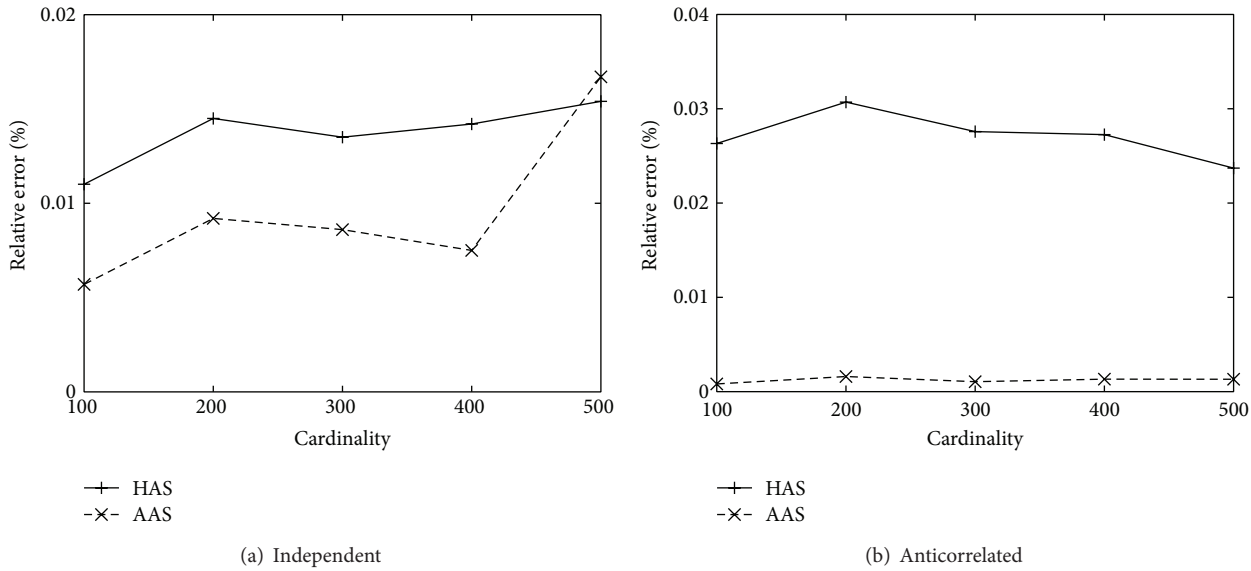
(a) Independent (b) Anticorrelated

Figure 9: Number of message versus cardinality.

using the mapping and filtering is proposed to reduce the communication cost of evaluating β-approximate skyline. Moreover, we extended definition of β-approximate skyline to β-hypothetical skyline and proposed the HAS algorithm whose skyline result consists of hypothetical tuples instead of real ones to further reduce the communication cost. Our experimental results show that both AAS and HAS are energy-efficient in evaluating approximate skylines in WSNs.

Notations

D: Dataset
D': Mapped dataset
n: Number of sensor nodes
d: Dimensionality of D
t: A tuple in D
$f(t)$: The mapped tuple of t
$t_i \succeq t_j$: t_i dominates t_j
$t_i \succ t_j$: t_i strictly dominates t_j
S = Skyline(): The skyline operator
$\widetilde{S}$ = ApprSkyline$_\beta$(): The β-approximate skyline operator
Dis($\widetilde{S}, S$): Distance between the surface of $\widetilde{S}$ and S
$l[k]$: Lower bound on dimension k
$u[k]$: Upper bound on dimension k
$r[k]$: Mapping range on dimension k.



(a) Independent

(b) Anticorrelated

Figure 10: Relative error versus cardinality.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This research was partially supported by the National Natural Science Foundation of China under Grant nos. 61472069 and 61100022 and the Fundamental Research Funds for the Central Universities under Grant no. N130404014.

References

[1] R. Cardell-Oliver, M. Kranz, K. Smettem, and K. Mayer, "A reactive soil moisture sensor network: design and field evaluation," *International Journal of Distributed Sensor Networks*, vol. 1, no. 2, pp. 149–162, 2005.

[2] W. Xue, Q. Luo, L. Chen, and Y. Liu, "Contour map matching for event detection in sensor networks," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 145–156, June 2006.

[3] S. Börzsönyi, D. Kossmann, and K. Stocker, "The skyline operator," in *Proceedings of the 17th International Conference on Data Engineering*, pp. 421–430, April 2001.

[4] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "TAG: a tiny aggregation service for ad-hoc sensor networks," in *Proceedings of 5th Symposium on Operating System Design and Implementation*, pp. 131–146, Boston, Mass, USA, December 2002.

[5] L. Q. Pan, J. Z. Li, and J. Z. Luo, "Approximate skyline query processing algorithm in wireless sensor networks," *Journal of Software*, vol. 21, no. 5, pp. 1020–1030, 2010.

[6] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang, "Skyline with presorting," in *Proceedings of the 19th International Conference on Data Ingineering*, pp. 717–719, March 2003.

[7] K. Tan, P. Eng, and B. Ooi, "Efficient progressive skyline computation," in *Proceedings of the 27th International Conference on Very Large Data Bases*, pp. 301–310, 2001.

[8] D. Kossmann, F. Ramsak, and S. Rost, "Shooting stars in the sky: an online algorithm for skyline queries," in *Proceedings of the 28th International Conference on Very Large Data Bases (VLDB '02)*, pp. 275–286, 2002.

[9] D. Papadias, Y. Tao, G. Fu, and B. Seeger, "An optimal and progressive algorithm for skyline queries," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 467–478, June 2003.

[10] W. Jin, J. Han, and M. Ester, "Mining thick skylines over large databases," in *Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pp. 255–266, 2004.

[11] D. Papadias, Y. Tao, G. Fu, and B. Seeger, "Progressive skyline computation in database systems," *ACM Transactions on Database Systems*, vol. 30, no. 1, pp. 41–82, 2005.

[12] C.-Y. Chan, H. V. Jagadish, K.-L. Tan, A. K. H. Tung, and Z. Zhang, "Finding k-dominant skylines in high dimensional space," in *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD '06)*, pp. 503–514, June 2006.

[13] C. Chan, H. Jagadish, K. Tan, A. Tung, and Z. Zhang, "On high dimensional skylines," in *Proceedings of the 10th International Conference on Extending Database Technology*, pp. 478–495, 2006.

[14] X. Lin, Y. Yuan, Q. Zhang, and Y. Zhang, "Selecting stars: the k most representative skyline operator," in *Proceedings of the 23rd International Conference on Data Engineering (ICDE '07)*, pp. 86–95, April 2007.

[15] K. Benouaret, D. Benslimane, and A. Hadjali, "On the use of fuzzy dominance for computing service skyline based on QoS," in *Proceedings of the IEEE 9th International Conference on Web Services (ICWS '11)*, pp. 540–547, July 2011.

[16] K. Benouaret, D. Benslimane, and A. Hadjali, "WS-Sky: an efficient and flexible framework for QoS-aware web service

selection," in *Proceedings of the IEEE 9th International Conference on Services Computing (SCC '12)*, pp. 146–153, June 2012.

[17] H. Chen, S. Zhou, and J. Guan, "Towards energy-efficient skyline monitoring in wireless sensor networks," in *Proceedings of the 4th European Conference on Wireless Sensor Networks*, pp. 101–116, 2007.

[18] J. Xin, G. Wang, L. Chen, X. Zhang, and Z. Wang, "Continuously maintaining sliding window skylines in a sensor network," in *Proceedings of the 12th International Conference on Database Systems for Advanced Applications*, pp. 509–521, 2007.

[19] J. Xin, G. Wang, and X. Zhang, "Energy-efficient skyline queries over sensor network using mapped skyline filters," in *Advances in Data and Web Management: Proceedings of the Joint 9th Asia-Pacific Web Conference, APWeb 2007, and 8th International Conference, on Web-Age Information Management, WAIM 2007, Huang Shan, China, June 16–18, 2007*, vol. 4505 of *Lecture Notes in Computer Science*, pp. 144–156, Springer, Berlin, Germany, 2007.

[20] W. Liang, B. Chen, and J. X. Yu, "Energy-efficient skyline query processing and maintenance in sensor networks," in *Proceedings of the 17th ACM Conference on Information and Knowledge Management (CIKM '08)*, pp. 1471–1472, October 2008.

[21] B. Chen and W. Liang, "Progressive skyline query processing in wireless sensor networks," in *Proceedings of the 5th International Conference on Mobile Ad-hoc and Sensor Networks*, pp. 17–24, December 2009.

[22] J. Xin, G. Wang, L. Chen, and V. Oria, "Energy-efficient evaluation of multiple skyline queries over a wireless sensor network," in *Proceedings of the 14th International Conference on Database Systems for Advanced Applications*, pp. 247–262, 2009.

[23] H. Shen, Z. Chen, and X. Deng, "Location-based skyline queries in wireless sensor networks," in *Proceedings of the International Conference on Networks Security, Wireless Communications and Trusted Computing (NSWCTC '09)*, pp. 391–395, Wuhan, China, April 2009.

[24] I.-F. Su, Y.-C. Chung, C. Lee, and Y.-Y. Lin, "Efficient skyline query processing in wireless sensor networks," *Journal of Parallel and Distributed Computing*, vol. 70, no. 6, pp. 680–698, 2010.