IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 9, 2016, accepted November 24, 2016, date of publication February 17, 2017, date of current version March 28, 2017.

Digital Object Identifier 10.1109/ACCESS.2016.2639558

Skyline Preference Query Based on Massive and Incomplete Dataset

YAN WANG, ZHAN SHI, JUNLU WANG, LINGFENG SUN, AND BAOYAN SONG
School of Information, Liaoning University, Shenyang 110036, China

Corresponding author: B. Song (bysong@lnu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61472072, Grant 61528202, Grant 61472169, and Grant 61501105, in part by the National Key Basic Pre-Research Program of China under Grant 2014CB360509, in part by the Foundation of Science Public Welfare of Liaoning Province in China under Grant 2015003003, and in part by the Fundamental Research Funds for the Central Universities under Grant N150404018.

ABSTRACT Personalized recommendation and the processing of real-time data exemplify the processing of massive data which in the field of Internet-of-Things (IoT) received a great extent of attention in recent literature. The incompleteness of massive data in the IoT is widespread. Obtaining personalized information from the incomplete data set is still puzzled by searching efficient and accurate methods at present. Skyline query is a widely used data processing method, especially in the field of multi-objective decision analysis and data visualization. To eliminate the negative effects on massive data processing in IoT, a novel skyline preference query strategy based on massive and the incomplete data set is proposed in this paper. This strategy simply separates and divides massive and incomplete data set into two parts according to dimension importance and executes skyline query, respectively. The strategy mainly resolves the problem of extracting personalized information from massive and incomplete data set and improves the efficiency of skyline query on massive and incomplete data set. First, this paper presents a skyline preference query strategy based on strict clustering and implements it on dimensions that have higher importance. Second, a skyline preference query strategy based on loose clustering is implemented on dimensions that have lower importance. Finally, integrating local skyline query results, this paper calculates global skyline query results by using information entropy theory. The efficiency and effectiveness of Skyline Preference Query (SPQ) algorithm have been evaluated in terms of response time and result set size through the comparative experiments with ISkyline algorithm and sort-based incomplete data skyline algorithm. A large number of simulation results show that the efficiency of SPQ algorithm is higher than that of other common methods.

INDEX TERMS Internet of things, incomplete data processing, skyline query, information entropy.

I. INTRODUCTION

The Internet of Things (IoT) and Cyber-Physical System (CPS) have made it possible for devices around the world to acquire information and store it, in order to be able to use it at a later stage [1], [2]. Currently, data is obtained by sensors and monitors in the field of IoT [3]–[5]. Due to the sensors and monitors failure, error and the existence of restrictions in acquiring actual data, misunderstanding data and missing values in datasets [6], the incompleteness of massive data is commonly observed in the field of Internet-of-Things [7], [8]. The dataset includes tuples have missing values in some of their dimensions, known as the incomplete data set. With the development and popularization of the Internet-of-Things, personalized recommendation aims to meet the needs of users becomes the hotspot of data processing in the field of Internet-of-Things. For example, according to the data from the smart bracelet, smart watches, and other wearable devices, different manufacturers can recommend their products for different users based on manufacturers preferences. The problem is that how we obtain the suitable information, which meets the users' needs, in incomplete and massive dataset? Skyline query [9] is a typical multi-objective optimization method to solve this problem, which plays a key role in decision-making, market analysis,

environmental monitoring, data mining, database visualization and econometrics [10]. Therefore, the skyline preference query processing is a new angle and a cut-in point for solving the problem of the massive incomplete data from Internet-of-Things and Cyber-Physical System. In the past, the incomplete data set was cleaned, repaired or processed by any pre-treatment (e.g., see [11]–[13]) before skyline query [14], [15]. However, pre-treatment consumes too much system resources and resulted in enormous errors in the repaired data, which leads to inaccurate outcomes. Further, for some timeliness problems, such as processing the data in influenza period, pre-processing these massive and timeliness data may lead to slow response and data invalid. In this paper, the pre-treatment stage of the traditional method is abandoned, and a skyline preference query strategy is proposed, which divided data set into two parts according to dimension importance: Skyline Preference Query Based on Massive and Incomplete Dataset (SPQ). The execution efficiency of skyline query on the massive and incomplete data set has been significantly enhanced and personalized data that satisfies users preference can also be obtained.

The major contributions of this paper are as follows:

- Our strategy gets rid of the pre-treatment stage that in general skyline query on incomplete data set, which reduces the response time of system.
- We divide incomplete dataset according to the importance degree of attributes. Skyline query based on strict clustering on the projection of attributes have higher importance degree can ensure accuracy and skyline query based on loose clustering on the remaining's of dataset can improve efficiency.
- We define tuple encoding, strict cluster encoding and inclusion relation as the basis of SPQ algorithm.
- We introduce two new algorithms, namely, SAVO algorithm and skyline preference query based on dominance degree which are executed on two projections of incomplete dataset respectively.
- We introduce a result selection strategy based on information entropy computation as a novel method designed specially for resolving the problem that the intersection of SSRS and RSRS is empty.
- We give experimental evidence that the SPQ algorithm is efficient and individual, especially for massive and high-dimensional dataset.

The remainder of this paper proceeds as follows: Section II provides an overview of the related work in the literature. Encoding and clustering strategies are illustrated in Section III. In Section IV, two skyline preference query strategies and a selection strategy based on information entropy are described. This is followed by simulation results and performance evaluation of SPQ presented in Section V. Finally, the paper is concluded in Section VI.

II. LITERATURE REVIEW

The probability distribution function of the data value is defined [16] in each data dimension. Based on the probability distribution function, the tuple of missing data is mapped to the tuple of complete data before their comparison. In fact, the missing values of incomplete data set are filled up in this method. But it can only be feasible in the case of the small data set. On the one hand, when applying it to the sparse data set, this method performs badly. On the other hand, this filling method cannot provide high-quality services to users, especially in the personalized recommendation problem that is related to user preference. In [17], a system model and SSQ (Subspace skyline query) algorithm are proposed to apply it on subspace skyline queries in a mobile distributed environment. Choosing one subspace of attributes per user preference, and then this method filters skyline query results by using an SQM-filtering algorithm and ε-filtering algorithm at this subspace. This algorithm can achieve satisfactory result when users estimate the domain of attributes accurately. If users don't know each attributes' value range, many usable data will be pruned before executing skyline query. So, this method cannot guarantee the accuracy of the skyline query. A bucket algorithm which redefines the tuples relationship in missing data set and the introduction of the virtual point and shadow skyline are proposed by Khalefa *et al.* [18]. Adding virtual point into buckets will destroy original bucket structure and if the virtual point doesn't dominate any points in the bucket, the number of comparison between tuples will increase drastically. Furthermore, this method must maintain and update shadow skyline at each bucket. Query cost grows exponentially when data is massive. Bharuka and Kumar [19] present a sort-based skyline algorithm, in which each dimension of the data is sorted and some top points in the sorted array domain the others so that the execute time and the number of pairwise comparisons between tuples both decrease. But this algorithm ignores the most practical case of involving incomplete data tuples where the points dominated by other points maybe domain the point that is processed now and that case will lead to adverse impaction of query accuracy. In (e.g., [20]–[22]), dynamic skyline query derived from the traditional skyline query are studied individually. The dynamic skyline query needs more user information that is users give a target point and this skyline query could return the information that user wanted.

III. CLUSTERING AND ENCODING STRATEGY

The dominance relation between tuples of a complete data set is transitive. For example, tuple p_i dominates p_j, p_j dominates p_k, so p_i dominates p_k. The missing dimensions and ambiguous dominance rules of tuples in incomplete dataset lead to cyclic dominance relation. Then, this cyclic dominance relation leads to empty skyline query result set. Suppose there only exist three tuples: p_i, p_j, p_k, $p_i = (4,^*, 2, 3)$, $p_j = (2, 3,^*, 3)$, $p_k = (^*, 2, 4, 3)$, p_i dominates p_j first and fourth dimensions, p_j dominates p_k at second and fourth dimensions, but p_i doesn't dominate p_k at any dimension. On the contrary, p_k dominates p_i at third and fourth dimensions. There exists the cyclic dominance relation and finally no skyline points are selected from the dataset. To solve the

problem of non-transitive and cyclic dominance relation of the incomplete data set, this paper proposes a clustering encoding strategy. After clustering, tuples in each cluster have same missing dimensions, and traditional rule of skyline query is applied to the other available dimensions. By utilizing this strategy, the dominance relation of tuples in the same cluster is transitive and eliminates the cyclic domination. In this paper, we add the user preference to traditional skyline query, i.e., the user is given the order of dimensions regarding their importance degree.

A. ENCODING STRATEGY

To cluster tuples, this approach encodes tuples firstly. Definition 1 gives the formal description of tuples' encoding on incomplete data set:

Definition 1 (Tuple Encoding): $\forall p'_i \in IS'(\text{or } p''_i), p'_i \cdot tuple\_code(\text{or } p''_i \cdot tuple\_code) = M_i, M_i = (m_1, m_2, \cdots, m_k)$; If $p'_i \cdot v_k(\text{or } p''_i \cdot v_k) =^*, M_i \cdot m_{ik} = 0$; If $p'_i \cdot v_k(\text{or } p''_i \cdot v_k) \neq^*, M_i \cdot m_{ik} = 1, k \in [1, \lambda]([\lambda + 1, d])$.

As shown above, IS' is the projection of top λ dimensions and IS'' is the remaining's. d denotes the dimension of the incomplete data set. p'_i is the projection of top λ dimensions and p''_i is the remaining's. M_i is tuple p_i's encoding, λ is dimension partition constant and $\lambda \in [1, d]$. Given a tuple $p = (1, 2, ^*, 3, 2)$, its tuple encoding is (1,2,0,1,1). According to tuple encoding, tuples who have same tuple encoding will be clustered into a cluster. As shown in Table 1, there is an incomplete data set, called IS. The incomplete data set is composed of seven tuples which have ten dimensions and "*" represents the missing value of a tuple. The data set has been sorted per dimension importance in non-ascending order and hence the order of dimensions at Table 1 is $D_1 > D_2 > \cdots > D_{10}$. Top λ dimensions are endowed with relative weight: 0.5,0.4,0.3,0.2,0.1, so are the remaining dimensions. That means the greater the relative weight, the greater the degree of preference for the user.

TABLE 1. The example of incomplete data.

	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10
P1	*3*	*3*	*	*2*	*5*	2	*	1	2	5
P2	*	*	3	5	*	*3*	*2*	*1*	*	*2*
P3	2	1	*	4	8	5	5	4	*	9
P4	10	4	2	*	5	*7*	*7*	*	*6*	*1*
P5	*	*1*	*1*	*2*	*5*	9	*	3	3	2
P6	3	*	*	9	8	*	*	*1*	*	*5*
P7	4	2	6	*	*	2	3	1	4	*

B. CLUSTERING STRATEGY

The clustering strategy that is designed specifically for this paper, which is based on the following considerations. First, the number of tuple encoding is within $[1, 2^d]$. With the increase in the number of dimensions, the number of tuple encoding will show exponential growth. The performance of skyline query will obviously degrade because the execute times is proportional to the number of clusters. In the process of skyline query, the dimensions have higher importance should get more consideration than the dimensions that have lower importance. To improve the efficiency of skyline query, this paper divides the incomplete data set into two parts regarding dimension importance. We will execute skyline preference query algorithm based on strict clustering and loose clustering at IS' and IS'' respectively.

Each cluster has a corresponding cluster encoding, and all the cluster encoding are stored in a set.

For strict clustering strategy, each tuple will be put in a cluster if its tuple encoding totally matches this cluster encoding. The following is the definition of strict cluster encoding:

Definition 2 (Strict Cluster Encoding): $\forall p'_i \in IS'$, if $\forall cc_j \in CS'$, let $cc_j \neq p'_i \cdot tuple\_code$, then $CS' = CS' \cup \{p'_i \cdot tuple\_code\}$.

CS' is an encoding set of strict clustering strategy, and cc_j represents a cluster encoding.

In contrast to strict clustering strategy, loose clustering strategy is that tuple encodings needn't exactly match to cluster encodings, in other words, if tuple encoding and cluster encoding satisfy inclusion relation, this tuple can be gathered in this cluster. The following is the definition of loose cluster encoding:

Definition 3 (Loose Cluster Encoding): In this definition, we firstly set that $d_1 = \lambda, d_2 = d - \lambda. \forall p''_i \in IS'', p''_i \cdot tuple\_code = M''_i, M''_i = (m_{ik}, m_{ik+1}, \cdots, m_{id})$, if $\exists cc_j \in CS''$ and $i \neq j, cc_j = (c_{jk}, c_{jk+1}, \cdots, c_{jd})$, for $\forall k \in [d_2, d], m_{ik} \leq c_{jk}$, p''_i will be included in cluster whose encoding is cc_j, p''_i satisfies inclusion relation with cc_j that denoted by $p''_i \rightarrow cc_j$, else if, $\exists cc_j \in CS''$ for $\forall k \in [d_2, d], m_{ik} \succ c_{jk}$ then $cc_j \rightarrow p''_i$ and this cluster encoding should be updated by $p''_i \cdot tuple\_code$ and that cc_j will be removed from CS''. A new cluster should be created when either there has no inclusion relation between $p''_i \cdot tuple\_code$ and cc_j or $CS'' = \emptyset$. $CS'' = CS'' \cup \{p''_i \cdot tuple\_code\}$ in all conditions except that p''_i will be included in cluster whose encoding is cc_j.

Among them, CS'' is an encoding set of loose clustering strategy. When a tuple satisfies multiple inclusion relation, it can be put in clusters where the cluster encoding corresponds to its tuple encoding. It is useful that tuples have higher similarity can be fully compared so that dominated tuples could be pruned early.

For a tuple p'_i in massive and incomplete data set, strict clustering rule and loose clustering rule are implemented on p'_i and p''_i, respectively. Utilizing Table 1 to illustrate the process of strict clustering and loose clustering: $p'_1 = (3, 3, ^*, 2, 5), p'_1 \cdot tuple\_code = (1, 1, 0, 1, 1), p'_3 = (2, 1, ^*, 4, 8), p'_3 \cdot tuple\_code = (1, 1, 0, 1, 1)$. In terms of strict clustering rule, p'_1 and p'_3 are added to same cluster. Clustering encoding is $cc_0 = (1, 1, 0, 1, 1)$ and $p'_4 = (10, 4, 2, ^*, 5), p'_4 \cdot tuple\_code = (1, 1, 1, 0, 1), p'_7 = (4, 2, 6, ^*, ^*), p'_7 \cdot tuple\_code = (1, 1, 1, 0, 0), p'_4 \cdot tuple\_code \neq p'_7 \cdot tuple\_code$.. p'_4 and p'_7

should be put in different clusters because their clustering encoding are $cc_1 = (1, 1, 1, 0, 1)$ and $cc_2 = (1, 1, 1, 0, 0)$. Now, $CS' = CS' \cup \{cc_1, cc_2\}$.

As for the process of loose clustering, supposed that there are four tuples: p''_1, p''_5, p''_6 and p''_7.

$p''_1 = (2,*,1,2,5), p''_1 \cdot tuple\_code = (1,0,1,1,1), p''_5 = (9,*,3,3,2), p''_5 \cdot tuple\_code = (1,0,1,1,1), p''_1 \rightarrow 10111$ and $p''_5 \rightarrow 10111$. But $p''_6 = (*,*,1,*,5), p''_6 \cdot tuple\_code = (0,0,1,0,1), p''_6 \rightarrow 10111$. And $p''_7 = (2,3,1,4,*), p''_7 \cdot tuple\_code = (1,1,1,1,0)$ there has no inclusion relation between p''_7 and cc_j. Consequently, p''_1, p''_5 and p''_6 are gathered into cluster 10111, p''_7 is put into cluster 11110. $CS'' = \{(1,0,1,1,1),(1,1,1,1,0)\}$.

IV. SKYLINE PREFERENCE QUERY STRATEGY

A. SKYLINE PREFERENCE QUERY STRATEGY BASED ON STRICT CLUSTERING

The skyline preference query strategy based on strict clustering performs in two steps. Firstly, this strategy implements strict clustering on tuples from IS' and then executes skyline preference query algorithm based on attribute value ordering on tuples that aren't dominated by others.

The processing of skyline preference query algorithm based on attribute value ordering can be divided into four phases.

Phases1: Sorting attributes in non-ascending order makes tuples which more likely dominate others to be processed prior. After sorting, each dimension will generate an array D_i, $i \in [1, \lambda]$ and$D_i[j] \geq D_i[j+1], j \in [1, |IS'|)$, $|IS'|$ is the number of tuple in IS'. If a tuple has missing value at i^{th} attribute, it will not be added to the array D_i. To save storage space, array D_i doesn't store complete tuple but its id. Setting a pointer ptr_i points to array D_i and all the tuples that not be dominated will be added to Candidate_Set. This method selects an array D_i randomly and then process the tuple pointed by pointer ptr_i. Each tuple in Candidate_Set has two values, one is the number of being processed that is denoted as processedCount and the other is the number of character '1' in a tuple encoding the number of non-missing attributes is denoted as dimCount.

Phases 2: For the currently selected tuple, there are three cases. Case 1: If tuple p' has never been processed and still be in the Candidate_Set, it can be compared with the tuple p'_j that there is no comparison between them even if p'_j has been dominated by tuple processed before. This is because p'_j maybe dominates p'; If there exists a tuple can dominate p', tuple p' should be removed from Candidate_Set. Case 2: If tuple p' has never been processed but is dominated by tuple processed before that p' will not be in Candidate_Set anymore. Therefore, tuple p' only compares with the tuples which still be in Candidate_Set and there is no comparison between them. In case 1 and case 2, any tuple is dominated by tuple p' will be removed from Candidate_Set. Case 3: If a tuple has been processed before, it will not compare to any tuple.

Special case: If there are two tuples, p'_i and p'_j, p'_i dominates p'_j only in few dimensions (less than the half number of dimensions), then p'_i will dominate p'_j regarding traditional skyline rule. However, considering user preference, the missing dimensions in tuples might have higher weights than those available dimensions. We should take those missing dimensions into consideration to ensure the accuracy of dominance relation between incomplete tuples. Judging the dominance relation of two tuples only through comparable dimensions is inaccurate. Especially, this paper defines the weak dominance relation among tuples with missing dimensions.

Definition 4: Weak Dominance Relation $\forall p'_i, p'_j \in IS'$, $i \neq j$, p'_i and p'_j the comparable dimension less than $\lceil \frac{\lambda}{2} \rceil$, in these comparable dimensions, if p'_i better than p'_j at not less than one dimension, not worse than p'_j, then p'_i weak dominates p'_j, which denoted as $p'_i \succ_* p'_j$.

If there is a weak dominance relation between two tuples p'_i and p'_j, it will need to compare the weight of the two tuples. If the weight of p'_i is higher than that of p'_j, it is considered that p'_i dominates the p'_j. In this paper, we present a tuple weight formula (1) as follows, which is based on the comprehensive consideration of the non-missing dimensions and the weights given by the user.

$$p'_i \cdot weight = \sum_{j=1}^{\lambda} p'_i \cdot v_j \times w_j \tag{1}$$

We still use the data in Table 1 to illustrate the calculation method of tuple weight. There are two tuples, $p'_2 = (*,*,3,5,*), p'_4 = (10,4,2,*,5)$. p'_2 and p'_4 can compare only at third dimension, and $p'_2 \cdot v_3 \succ p'_4 \cdot v_3$. So, they satisfy the weak dominance relation and need to calculate tuple weight. $p'_2 \cdot weight = 3*0.3+5*0.2 = 1.9 p'_4 \cdot weight = 10*0.5 + 4*0.4 + 2*0.3 + 5*0.1 = 7.7$ p'_2 doesn't dominate p'_4 owing to $p'_2 \cdot weight \prec p'_4 \cdot weight$.

Phase 3: When the number of comparisons of tuple p equals to $p \cdot dimCount$, which is the number of non-missing dimensions of p. Then p will be removed from the Candidate_set to the strict skyline result set SSRS.

Phase 4: When the candidate set is empty or all tuples are processed at least once, the rest of tuples in Candidate_Set are push back to the strict skyline clustering result set SSRS.

As is shown below, this is the algorithm to judge the dominance relation between two tuples.

The pseudo code of skyline preference query algorithm based on attribute value ordering is as follows:

We also describe the skyline preference query algorithm based on attribute value ordering by taking an example of the incomplete data set from Table 1. The result of IS' that is sorted by attribute value is shown in Table 2. p_4 dominates p_1, p_2 weak dominates p_4. However, $p_2 \cdot weight \prec p_4 \cdot weight$, so p_2 could not dominate p_4. There is no dominance relation between p_4 and p_5. Similarly, there is also no dominance relation between p_4, p_6 and p_7. After the first iteration, $p_4 \cdot processedCount = 1$. p_1 and p_5 are removed from Candidate_Set. Then, ptr_1 also pointers to the first tuple p_4

Algorithm 1 Dominate (p,q)

1. *Input*: tuple p,q
2. *Output*: Integer value
3. **If** p>q
4. **Return** 1;
5. **Else if** p>$_*$q
6. **If** $p \cdot weight \succ q \cdot weight$
7. **Return** 1;
8. **Else if** q>$_*$p
9. **If** $q \cdot weight \leq p \cdot weight$
10. **Return** 1;
11. **Return** 0

TABLE 2. The result of sorted dimensions.

D1	D2	D3	D4	D5
P4	P4	P7	P6	P3
P7	P1	P2	P2	P6
P1	P7	P4	P3	P1
P6	P3	P5	P1	P4

of array D2. Since p_4 has been processed before, it need not compare with any tuple. In the end of the second iteration, $p_4 \cdot processedCount$ is 2. In the third iteration, the tuple of array D3, p_7 is processed and it should compare with $p_1 - p_3$, p_5 and p_6. Actually, p_7 is not dominated by p_1 and p_5 which have been removed from the Candidate_Set at the first iteration. p_7 weak dominates p_2 and $p_7 \cdot weight > p_2 \cdot weight$, so p_7 dominates p_2. Similarly, p_7 dominates p_3 and p_6. By the end of this iteration, p_2, p_3 and p_6 are removed from Candidate_Set. When the iteration is carried out to the fourth time, p_6 should be processed but it has not been at Candidate_Set hence it fulfils the condition of phase 2 of SAVO algorithm. It needs to compare with tuples still be in Candidate_Set and they should have never been compared with each other before. Currently, Candidate_Set only have two tuples, p_4 and p_7, which have been compared with p_6. In the end, $p_6 \cdot processedCount$ is 1. Until the twelfth iteration,$p_7 \cdot processedCount = p_7 \cdot dimCount = 3$, so it can be removed from Candidate_Set to SSRS. By the eighteenth iteration, p_5 is processed and there are no remaining tuples has not been processed. Therefore, SAVO algorithm reaches to its end. We attain two tuples, p_4 and p_7, which will become candidates' skyline points via strict clustering strategy and skyline preference query algorithm based on attribute value ordering.

B. SKYLINE PREFERENCE QUERY STRATEGY BASED ON LOOSE CLUSTERING

According to the loose clustering rule described in Section III, we can attain the loose clusters on IS''. But tuples in each cluster might have no dominance relation. These tuples will also be the skyline query result return to users. Thus, users need to select what they want because

Algorithm 2 Skyline Preference Query Algorithm Based on Attribute Value Ordering SAVO

1. *Input*: IS'
2. *Output*: SSRS
3. initialize Candidate_Set $= IS'$, $SSRS = \emptyset$, iter=1
4. **Foreach** dimension $s_i \in IS'$
5. This step sorts the attribute value at s_i by non-ascending order and stores corresponding tuple id in array D_i. If tuples have the same attribute value, the smaller tuple id, the tuple has more priority to be stored.
6. The pointer ptr_i in array D_i is initialized by 1.
7. **Endfor**
8. **Foreach** $p' \in IS'$
9. Initialize $p' \cdot processedCount = 0$;
10. $p' \cdot dimCount$=the number of dimensions that values are non-missing;
11. **Endfor**
12. **While** Candidate_Set≠∅ and at least one tuple has not been processed
13. nextDim=iter%λ;
14. $P = D_{nextDim}[ptr_{nextDim}]$
15. $P \cdot isDominated$=false;
16. **If** $P \cdot processedCount = 0$
17. **If** If $P \in$ Candidate_Set
18. **For** each q∈ (IS'-Candidate_Set)
19. **If** (P has never compared with q)
20. **If** (***dominate***(P,q)==0)
21. P *isDomainated* =true;
22. **Endif**
23. **Endif**
24. **Endfor**
25. **Endif**
26. **If** $P \cdot isDominated$==false
27. **Foreach** (c in Candidate_Set)
28. **If** (there has no comparison between p and c before)
29. **If**(***dominate***(P,c)==0)
30. $P \cdot isDominated$=true;
31. **Else**
32. Remove c from Candidate_Set;
33. **Endif**
34. **Endfor**
35. **Endif**
36. **If** $P \cdot isDominated$==true and P in Candidate_Set
37. Remove P from Candidate_Set;
38. **Endif**
39. **Endif**
40. $P \cdot processedCount++$;
41. **If** (P in Candidate_Set and $P \cdot processedCount == P \cdot dimCount$)
42. Remove P from Candidate_Set to SSRS;
43. **Endif**
44. Iter++;
45. $ptr_{nextDim}++$;
46. **Endwhile**
47. Return SSRS;

tuples in the result set may not meet the needs of users.

Considering this problem, we take the user preference as an important factor in the decision of dominance. A calculation method of dominance degree is proposed to reduce the redundancy of skyline query result set, which differs from most traditional methods. The dominance relation of tuples with user preference is determined by the degree of dominance between them. The following is the definition of the degree of dominance:

Definition 5: The Degree of Dominance

The difference of attribute value of any two tuples can be regarded as their dominance distance. The degree of dominance between any two tuples at the same cluster can be recorded as the sum of the product of dominance distance and weight on comparable dimensions. $w_{\lambda+1}, w_{\lambda+2}, \ldots, w_d$ are the weights of dimensions. The missing values of each dimension of tuple p_i are denoted by v_{ij}, $j \in [\lambda+1, d]$. The degree of dominance between any two tuples p_i and p_j is $domain_{i,j} = \sum_{k=d_2}^{d} w_k^*(v_{ik} - v_{jk})$, wherein k represents the k^{th} comparable dimension of a tuple. If $domain_{i,j} > 0$, then $p_j < p_i$. If $domain_{i,j} < 0$, $p_j > p_i$. If $domain_{i,j} = 0$, there are no dominance relation between p_i and p_j.

In each cluster, we calculate the degree of dominance of any two tuples. After the eliminating tuples that are dominated by others, remaining tuples at clusters will be added into loose clustering skyline query result set, RSRS.

C. RESULT SELECTION STRATEGY BASED ON INFORMATION ENTROPY COMPUTATION

In Section IV, we have obtained strict clustering result set SSRS and loose clustering result set RSRS. If the intersection of SSRS and RSRS is not empty, the intersection will be the global skyline query result set. Otherwise, we should calculate tuples' information entropy in SSRS and RSRS, respectively. According to the concept of information entropy, the greater the uncertainty of variables, the greater the entropy and the more information we need to make it clear. The more orderly a system is, the lower the information entropy is. On the contrary, the more chaotic a system is, the higher the information entropy is. Therefore, the information entropy can also be said to as a measure of the degree of order in a system. A data tuple can also be regarded as a system with many of factors. The attributes of a tuple are the influence factor of this system. The number of missing values in a tuple represents the uncertainty of the influencing factors in this specific system. The more uncertain factors (i.e., more missing values) it has, the more confusion and disorder it has. Similarly, this tuple will possess greater information entropy. Thus, a tuple that has the smaller information entropy is dominated by a tuple that has greater information entropy. The calculation formula of information entropy is given:

$$E(p_i) = \sum_{k=1}^{n} \ln(h' + 1) \tag{2}$$

TABLE 3. The result of loose clustering.

Cluster 10111	Cluster 11101	Cluster 11011	Cluster 11110
P1(2,*,1,2,5)	P2(3,2,1,*,2)	P4(7,7,*,6,1)	P7(2,3,1,4,*)
P5(9,*,3,3,2)	P3(5,5,4,*,9)		
P6(*,*,1,*,5)	P6(*,*,1,*,5)		

$E(p_i)$ presents the information entropy of tuple p_i, h' represents the normalized value of the attributes of a tuple, and n is the number of dimension of a tuple.

After the calculation of information entropy of tuples from SSRS and RSRS, we can provide the global skyline query result set for users.

Combined with definition 2 and definition 3 in Section III, this paper applies skyline preference query strategy based on loose clustering on incomplete data from Table 1. The result is shown in Table 3:

For Cluster 10111: $domain_{1,5} = 0.5^*(2-9)+0.3^*(1-3)+0.2^*(2-3)+0.1^*(5-2) = -4 < 0$, so $p_5 \succ p_1$; $domain_{5,6} = 0.3^*(3-1)+0.1^*(2-5) = 0.3 > 0$, so $p_5 \succ p_6$; For Cluster 11101: $domain_{2,3} = 0.5^*(3-5)+0.4^*(2-5)+0.3^*(1-4)+0.1^*(2-9) = -3.8 < 0$, $p_3 \succ p_2$; Similarly, $domain_{2,6} < 0$, $p_6 \succ p_2$; $domain_{3,6} > 0$, $p_3 \succ p_6$; Finally, we can obtain p_4, p_7. $RSRS = \{p_5, p_3, p_4, p_7\}$, $SSRS = \{p_4, p_7\}$. The global skyline preference query result set is $\{p_4, p_7\}$.

V. EXPERIMENT AND ANALYSIS

A. EXPERIMENTAL ENVIRONMENT

The experimental environment of this paper is a single PC machine, and PC machine is configured as Intel(R) Xeon(R) 3.4GHz, 16GB memory. The programming environment is Eclipse IDE, the programming language is Java and the code is compiled by JDK 1.8.0_91.

For evaluating the performance of the proposed approach, we have used both synthetic and real world data sets. We use standard data generation tool to generate a complete data set and then generate the incomplete data set with a random function. NBA [23] is real world data set, which consists of 21962 tuples and statistics about skills of basketball players during regular season in the period 1946-2009. Each tuple has 23 attributes while the 17 attributes of those are comparable, which contains field goal percentage, total points, rebound, assists and so forth on Steals, Blocks, and Turnovers were not recorded in the NBA until 1973, so we have used the data in the period of 1973-2009, which totally contains 16919 tuples. Owing to the completeness of NBA data set, we removed values from it to generate 20% incomplete data set.

Each dimension of the data is independent of each other and is in the Gauss distribution. The number of records of the experimental data set is 100K and the tuple dimension is 16. The weight of each dimension is 0.5,0.4,0.3,0.2,0.1,0.5,0.4,0.3,0.2,0.1 and the data missing rate is 20%. The parameter t in the ISkyline algorithm is set

to 100. Owing to a large amount of data, we divide the data into slices, and execute the algorithm in parallel, which can improve the efficiency of the algorithm.

Two indexes evaluate the performance of our algorithm: response time and the size of the result set. This paper puts forward the SPQ algorithm, ISkyline algorithm in the literature [11] and sort-based incomplete data skyline (SIDS) algorithm in the literature [12] for comparison. For evaluating the performance of obtaining personalized information of SPQ algorithm, this paper has designed an experiment in NBA data set, which compares the accuracy of result of SPQ algorithm with that of CDskyline algorithm proposed in literature [7]. We set different weights for each dimension of the NBA data set to obtain different skyline result.

B. EXPERIMENTAL RESULTS AND ANALYSIS

1) EFFECT OF DATA SET SIZE ON PERFORMANCE OF SKYLINE ALGORITHM

The main analysis of this experiment is that the execution time changes with the data set size. From Fig. 1, as the increase of tuples in the data set, the execution time of ISkyline algorithm show exponential growth approximately, while the SIDS algorithm and SPQ algorithm execution time account for about 10% in that of ISkyline algorithm. This is because the ISkyline algorithm introduces the virtual point and shadow skylines. With the increasing number of tuples, the number of barrels also increases, which leads to a lot of unnecessary comparisons between tuples from this bucket and other buckets. This consumes execution time a lot. In this paper, the initial execution time of SPQ algorithm is close to that of SIDS algorithm, and the execution time of SPQ algorithm is gradually lower than the execution time of the SIDS algorithm with the increase of the size of data set. SIDS algorithm needs to be performed in all dimensions, while the SPQ algorithm only needs to be performed in the λ dimension and with the increase of the size of the data set, the sorting time is raised.

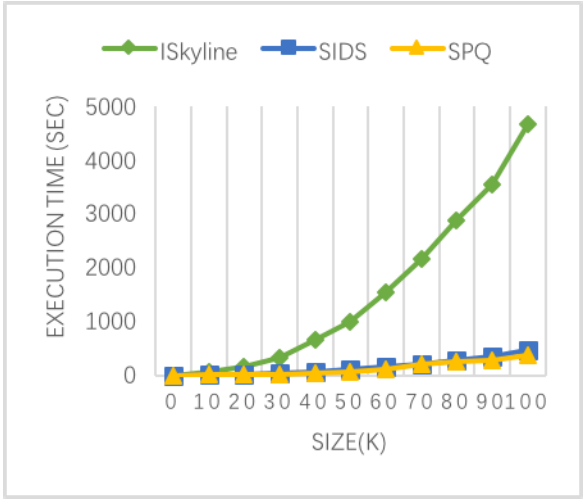

FIGURE 1. The effect of data set on execution time.

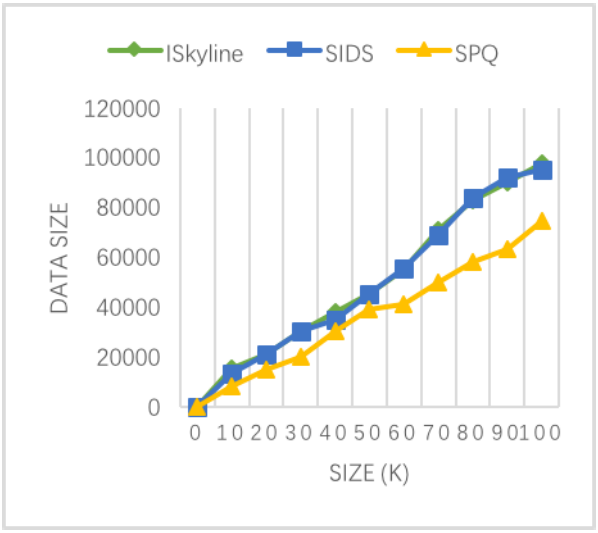

FIGURE 2. The effect of data size on result set size.

The size of the result set in this experiment is analysed with the change of data set size. The experimental result is shown in Fig. 2. The low number of tuples in result set that produced by SPQ algorithm is because SPQ algorithm takes user preference into consideration, which could decrease the number of extra tuples. However, with the rise of the size of incomplete data set, an increasing number of redundant results that do not meet the users' demand will be produced by implementing ISkyline algorithm and SIDS algorithm. This is because these two algorithms do not take user preference into consideration.

2) EFFECT OF THE DIMENSION OF DATA SET ON THE PERFORMANCE OF SKYLINE ALGORITHM

This experiment mainly analyses the execution time of algorithms along with the change of the number of dimension of the incomplete data set. The change interval of dimension in the experimental data set is [2,16], and the data set contains 100k records. From Fig. 3, we can intuitively see the slight difference between these three algorithms when the dimension of the experimental data set is changing from 2 to 10, while the execution time of ISkyline algorithm is far greater than the other two algorithms when the dimension of experimental dataset changes from 12 to 16. This is because as the dimension increases, the buckets of ISkyline algorithm grow exponentially, 2^n(n is the number of dimension of data). Due to an exponential growth of the number of buckets, comparisons between tuples show exponential growth. For the SIDS algorithm, the data dimension is increased, and the number of dimension ordering also arises. SPQ algorithm only executes sorting algorithm in the previous λ dimensions, while we calculate the degree of dominance of all dimensions on the back of the data set. The calculation time of the degree of dominance is slightly lower than the time of sorting dimensions.

This experiment analyses the size of the result set with the change of dimension of the incomplete data set. As can be

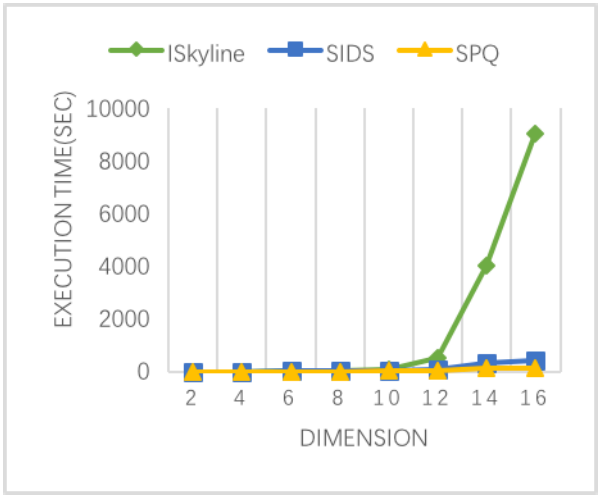

FIGURE 3. The effect of data dimensions on execution time.

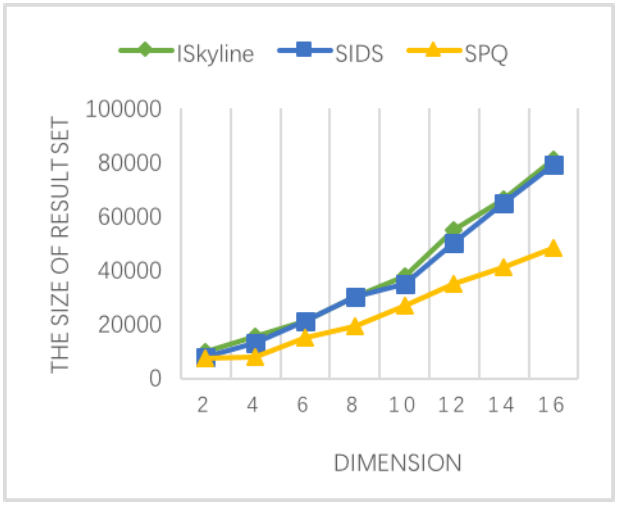

FIGURE 4. The effect of data dimension on the size of result set.

seen from Fig. 4, with the increase of the data dimensions, the growth rate of the result set of ISkyline algorithm and SIDS algorithm become larger than SPQ algorithms. With the increase of tuple dimension, dominance relation between tuples is more difficult to distinguish. This is because that the most missing dimensions, so the more incomparable tuples. According to the theory of skyline query, these tuples will be added to the result set. The SPQ algorithm proposed in this paper takes user preferences into account, and the tuple's weights are calculated in the process of the skyline, which makes the data tuple in the result set to be in line with the user requirements. Therefore, as the dimension increases, the result set get from the SPQ algorithm is smaller than that of other two algorithms. Gradually, the advantage of SPQ algorithm emerges.

3) THE EFFECT OF DATA MISSING RATE ON THE PERFORMANCE OF SKYLINE ALGORITHM

This experiment mainly analyses the effect of the data missing rate on the algorithm execution time. Fig. 5 and Fig. 6

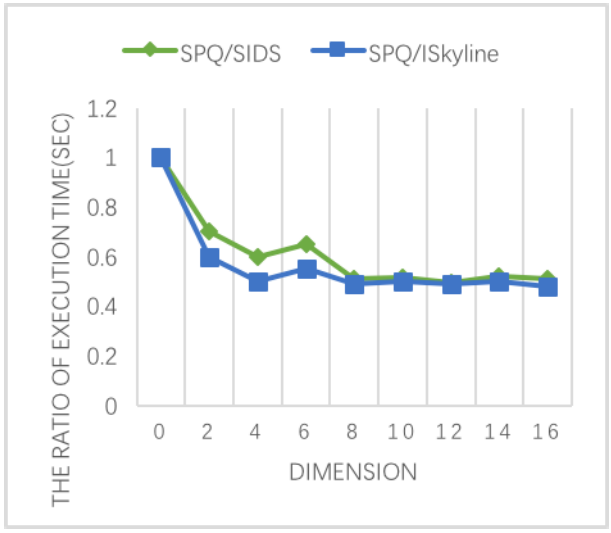

FIGURE 5. The effect of 10% missing data on execution time.

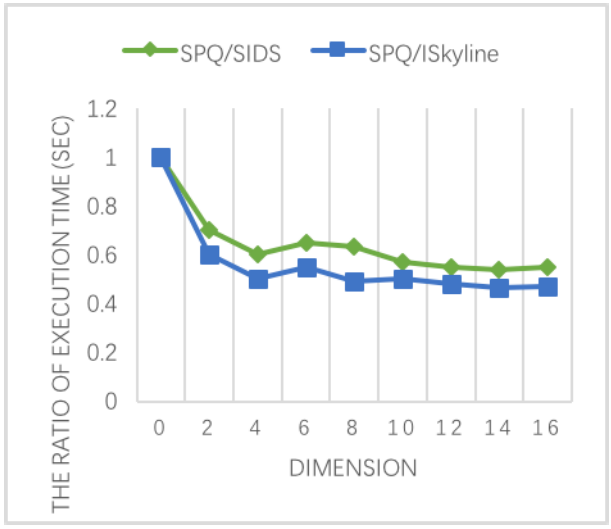

FIGURE 6. The effect of 50% missing data on execution time.

show that the ratio of SPQ algorithm and the other two algorithms in data missing rate is 10% and 50%, respectively. As can be seen from the graph, the reduction of the ratio of execution time explains the fact that SPQ algorithm is better than the SIDS and ISkyline algorithms. SPQ algorithm does not execute Bucket algorithm on all dimensions but performs strict clustering strategy on the dimensions have higher importance and performs loose clustering strategy on the other dimensions. It is the method that can reduce the time consumption. And after strict clustering, many of the tuples that dominated by others are removed from clusters. Compared with the SIDS algorithm, which sorts tuples attributes value directly, our method can cut down the time cost of ordering which executes on many tuples.

The data missing rate affects the execution time of SPQ algorithm not so much, as shown in two figures above. Before executing skyline query, SPQ algorithm has removed many tuples that dominated by others via clustering strategy so that

TABLE 4. The result of SPQ algorithm.

Desired Position	Preference	Player Name	Actual position
defender	assists>steals>total points>rebound>blocked shots	John Stockton	defender
		Michael Jordan	defender
		Allen Iverson	defender
		Kobe Bryant	defender
		World B. Free	defender
		Ervin Johnson	defender
		Allan Houston	defender
		Isaiah Rider	defender
		Sam Cassell	defender
		Eddie Jones	defender
center player	rebound>blocked shots>total points>assists>steals	David Robinson	center player
		Patrick Ewing	center player
		Bob McAdoo	center player
		Robert Parish	center player
		Roy Tarpley	center player
		Steve Johnson	center player
		Elden Campbell	center player
		Blair Rasmussen	center player
		Stanley Roberts	center player
		Marc Jackson	center player

TABLE 5. The result of CDskyline algorithm.

Desired Position	Preference	Player Name	Actual position
defender	assists>steals>total points>rebound>blocked shots	John Stockton	defender
		Mitch Richmond	defender
		Sam Cassell	defender
		Ervin Johnson	defender
		Jeff Malone	defender
		Chris Mullin	striker/defender
		Allan Houston	defender
		Isaiah Rider	defender
		Allen Iverson	defender
		Eddie Jones	defender
center player	rebound>blocked shots>total points>assists>steals	Wilt Chamberlain	center player
		Dennis Rodman	striker
		Bob McAdoo	center player
		Robert Parish	center player
		Roy Tarpley	center player
		Elvin Hayes	striker/center player
		George McGinnis	stricker/center player
		Blair Rasmussen	center player
		Stanley Roberts	center player
		Marc Jackson	center player

the comparison time between incomparable tuples decreases, and then data missing rate won't influence the execution time.

4) THE COMPARISON OF THE EFFECTS OF PREFERENCE SKYLINE QUERIES

This experiment mainly verifies the effect of the SPQ algorithm preference query. Assuming a user of an application is a basketball coach, he wants to recruit players in different positions on the team. In this experiment, we choose 5 of 17 skills from NBA data set as player criteria, which are total points, rebound, assists, steals and blocked shots. When he needs to recruit a defender, he might think that the importance of these 5 dimensions is: assists>steals>total points>rebound>blocked shots; he needs to recruit a centre player, he might think that the importance of these 5 dimensions is: rebound>blocked shots>total points>assists>steals. We give the relative weights of these 5 dimensions, which are sorted by importance: 0.8, 0.6, 0.5, 0.35, 0.25 while the relative weights of remaining's are 0.1 for the aim of easy calculation. Our experiment executes skyline preference query based on strict cluster strategy on the projection of five dimension of NBA data set and executes skyline preference query based on loose cluster strategy on the projection of remaining's of NBA data set, respectively.

After the execution of SPQ algorithm and CDskyline algorithm per the two dimension preferences above, we can obtain the query results about ten defenders and ten center players that are shown in Table 4, 5.

As shown in Table 4, when the user considers the degree of importance is: assists>steals>total points>

rebound>blocked shots, the result of SPQ algorithm consists of famous defenders such as Michael Jordan, Allen Iverson and Kobe Bryant. When the user considers the degree of importance is: rebound>blocked shots>total points>assists>steals, the result of SPQ algorithm contains the centre players that meet user's need. It does indicate that the result of executing SPQ algorithm with various user preferences on same data set could meet user's needs.

We can see from the data in Table 5 that the result of CDskyline algorithm cannot totally meet user's need. This is because CDskyline ignore the negative effects of missing values in incomplete dataset, which only calculates the non-missing values in each dimension in the procedure of skyline query. Although this algorithm can choose some defenders, it omits the excellent defenders, such as Kobe Bryant. Similarly, some of the center players were selected, but this algorithm also mistakenly elected few strikers. Thus, the accuracy of CDskyline is low. SPQ algorithm takes missing values into consideration, which calculates tuples weights if a tuple has a few missing values weakly dominates other tuple in few dimensions. This method, to some extent, ensures the accuracy of dominance between tuples and then obtains the personalized information that meet user's needs.

VI. CONCLUSIONS

Regarding the incomplete and missing data in the field of Internet-of-Things, this paper studies the personalized recommendation which combines the massive data of the Internet-of-Things with user preference. We propose a skyline preference skyline strategy based on massive and incomplete data set, including encoding and clustering strategy, two strategies of skyline preference query based on dimension importance clustering and a selection strategy based on information entropy theory. As far as practical implementations, we design a comparative experiment with ISkyline algorithm and SIDS algorithm about three aspects: data size, data dimension, and data missing rate. Experimental results based on synthetic data sets manifest that SPQ algorithm is competitively more efficient and positive in results accuracy than other algorithms.

REFERENCES

[1] D. Gil, A. Ferrández, H. Mora-Mora, and J. Peral, "Internet of Things: A review of surveys based on context aware intelligent services," *Sensors*, vol. 16, no. 7, p. E1069, Jul. 2016.
[2] H. Song, D. Rawat, S. Jeschke, and C. Brecher, *Cyber-Physical Systems: Foundations, Principles and Applications*. Boston, MA, USA: Academic, 2016, p. 514.
[3] G. G. Zhang, Y. Bi, C. Li, Y. Zhang, and C. Zeng, "Security processing model research based on massive loT data," *J. Chin. Comput. Syst.*, no. 9, pp. 2090–2094, Sep. 2013.
[4] L. Zhang, Y. Wang, B. Y. Song, X. Li, and X. Hao, "Geometry-based spatial skyline query in wireless sensor network," in *Proc. 11th Web Inf. Syst. Appl. Conf.*, Tianjin, China, Sep. 2014, pp. 27–32.
[5] Y. Wang, B. Y. Song, J. L. Wang, L. Zhang, and L. Wang, "Geometry-based distributed spatial skyline queries in wireless sensor networks," *Sensors*, vol. 16, no. 4, p. 454, Apr. 2016.
[6] Y. Gu, G. Yu, X. J. Li, and Y. Wang, "RFID data interpolation algorithm based on dynamic probabilistic path-event model," *J. Softw.*, no. 3, pp. 438–451, Mar. 2010.
[7] Y. Wang, B. Yin, G. H. Liu, B. Y. Song, and J. L. Wang, "Skyline query of massive incomplete data based on combinational dimensions," *J. Frontiers Comput. Sci. Technol.*, no. 4, pp. 495–503, Sep. 2016.
[8] S. Jeschke, C. Brecher, H. Song, and D. Rawat, *Industrial Internet of Things*. Switzerland: Springer, 2017, p. 715.
[9] L. L. Ding, J. C. Xin, G. R. Wang, and S. Huang, "Efficient skyline query processing of massive data based on Map-Reduce," *Chin. J. Comput.*, vol. 34, pp. 1786–1796, Oct. 2011.
[10] S. Borzsonyi, D. Kossmann, and K. Stocker, "The skyline operator," in *Proc. 17th ICDE*, 2001, pp. 421–430.
[11] J. J. Cao, X. C. Diao, S. Chen, and Y. Z. Shao, "Data cleaning and its general system framework," *Comput. Sci.*, vol. 39, pp. 207–211, Nov. 2011.
[12] Y. H. Lin, C. H. Zhang, and J. Liu, "Realization of data cleaning based on editing rules and master data," *Comput. Sci.*, vol. 39, pp. 174–176, Nov. 2012.
[13] W. Chen and Q. L. Ding, "Cleaning method for incomplete data in data cleaning," *Microcomput. Appl.*, no. 2, pp. 44–45, Feb. 2005.
[14] X. J. Wei, J. Yang, C. P. Li, and H. Chen, "Skyline query processing," *J. Softw.*, vol. 19, pp. 1386–1400, Jun. 2008.
[15] L. Zhu, J. H. Guan, and S. G. Zhou, "Review of skyline computing research," *Comput. Eng. Appl.*, no. 6, pp. 160–165, Feb. 2008.
[16] Z. Zhang, H. Lu, B. C. Ooi, and A. K. Tung, "Understanding the meaning of a shifted sky: A general framework on extending skyline query," *VLDB J.*, vol. 19, no. 2, pp. 181–201, Feb. 2010.
[17] Y. Y. Li, Z. Y. Li, M. X. Dong, W. Y. Qu, C. Q. Ji, and J. F. Wu, "Efficient subspace skyline query based on user preference using mapreduce," *Ad Hoc Netw.*, vol. 35, pp. 105–115, Nov. 2015.
[18] M. E. Khalefa, M. F. Mokbel, and J. J. Levandoski, "Skyline query processing for incomplete data," in *Proc. 24th ICDE*, 2008, pp. 556–565.
[19] R. Bharuka and P. S. Kumar, "Finding skylines for incomplete data," in *Proc. 24th ADC*, 2013, pp. 109–117.
[20] L. Zhang, P. Zou, Y. Jia, and L. Tian, "Continuous dynamic skyline queries over data stream," *J. Comput. Res. Develop.*, vol. 48, pp. 77–85, Jan. 2011.
[21] K. Ahmed, N. S. Nafi, and M. A. Gregory, "Enhanced distributed dynamic skyline query for wireless sensor networks," *J. Sens. Actuators Netw.*, vol. 5, no. 1, p. 2, Mar. 2016.
[22] Y. Park, J. K. Min, and K. Shim, "Parallel computation of skyline and reverse skyline queries using mapreduce," *Proc. VLDB Endowment*, vol. 6, no. 14, pp. 2002–2013, Sep. 2013.
[23] *Databasesports Website*. [Online]. Available: http://database basketball.com

YAN WANG received the Ph.D. degree in computer software and theory. She is currently an Associate Professor with the Department of Information, Liaoning University.

ZHAN SHI is currently pursuing the master's degree with Liaoning University. Her research interest is massive data query technology.

JUNLU WANG received the M.S. degree from Liaoning University in 2014. He is currently an Assistant Engineer with Liaoning University. His research interests include database theory and techniques, big data processing techniques, and massive data processing techniques. He is a member of CCF.

LINGFENG SUN is currently pursuing the master's degree with Liaoning University. Her research interest is massive data management.

BAOYAN SONG received the Ph.D. degree in computer software and theory from Northeastern University in 2002. She is currently a Professor with Liaoning University. Her research interests include database theory and techniques, RFID event stream processing techniques, big data management, and graph data management. She is a Senior Member of CCF.