Keyword-Matched Data Skyline in Peer-to-Peer Systems

Khaled M. Banafaa, Ruixuan Li*, Kunmei Wen, Xiwu Gu, and Yuhua Li

School of Computer Science and Technology,
Huazhong University of Science and Technology,
Wuhan, Hubei 430074, P.R. China
kbanafaa@smail.hust.edu.cn,
{rxli,kmwen,guxiwu,idcliyuhua}@hust.edu.cn

Abstract. Data and storage management is turning to distributed due to the huge increase in data volumes. To satisfy users' requirements and preferences, advanced query operators, such as skyline, have been introduced and implemented. Skyline offers users with interesting objects, which has been explored in centralized, distributed and peer-to-peer (P2P) systems. However, keyword-matched skyline has not been considered in distributed and P2P systems. This paper introduces keyword-matched data skyline algorithms in P2P systems. Differing from other operators, skyline algorithms are devised to exploit its properties to reduce traversed peers for a query. By partitioning data space and using distributed hash tables (DHTs) and Bloom filters, we design new algorithms, Nk-sky and Ck-sky, to reduce the required traversed peers to answer keyword-matched data skyline queries. We apply the algorithms on Chord as an example of DHT overlay P2P systems. Experimental results show a significant reduction of traversed peers with the *Cover-set tuples* algorithm Ck-sky.

Keywords: Peer-to-peer system, skyline query, keyword-matched skyline.

1 Introduction

With the fast growing and huge volumes of data in the current Internet environment, advanced queries in distributed systems have been introduced, studied and designed. Skyline operator introduced by Börzsönyi in [1] is an example of such advanced queries. It recommends some interesting data objects to the users. The interesting objects are dominating objects that a user would be more interested in than the other objects. On the other hand, the processing limitations of centralized systems lead to distributed system designs. Different distributed systems have been suggested and used to solve the problems efficiently. In peer-to-peer

* Corresponding author.

W. Meng et al. (Eds.): DASFAA 2013, Part I, LNCS 7825, pp. 71–85, 2013.
© Springer-Verlag Berlin Heidelberg 2013

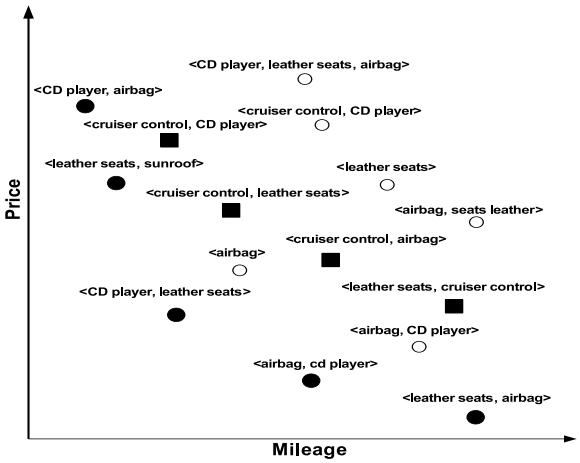

Fig. 1. An example of keyword-matched skyline

(P2P) systems, distributed hash tables (DHTs) have been discovered; and different network overlays have been designed (e.g. Content Addressable Network (CAN) [2], Chord [3] and Harmonic Ring (HRing) [4]). A query nature may result in a preference of network overlays to others. Skyline queries, like others, have been studied in different distributed systems and P2P system overlays (e.g. CAN as in [5], and BAlanced Tree Overlay Network (Baton) as in [6][7]).

A skyline point is a point that is not dominated by any other point in all dimensions. In general, the domination in one dimension is the user preference in that dimension (e.g. cheaper, lower mileage, and shorter distance). As an example, a user may be interested in buying a cheap used car with low mileage as shown in Fig. 1. The skyline query will return the black-filled rounded points in Fig. 1. However, a user may be interested in skyline for only points with some features. For example, a dealer may have many used cars with different features. Some cars come with cruiser controls, cd players, and/or airbags etc. A user, who is interested only in cars with cruiser control, may not be interested in the results that do not consider their preferences. The black-filled rectangular points shown in Fig. 1 are what that user expects. Another example may come from a user who is only interested in a restaurant with some dishes (e.g. sushi, seafood). Some restaurants may serve sushi but some restaurants may not. Thus, the traditional algorithms may result in skyline of no interest to the user. A skyline for only restaurants that serve sushi is what a user needs to see. These types of queries are called keyword-matched skyline queries.

Another example, consider some online scientific data analysis system where different participants publish their findings and use others' findings. Each participant may focus on different parts of the experiments. Keywords-matched skyline can help such scientists identify outstanding data and results of their interests. There could be large numbers of keywords-matched skyline queries triggered by

different scientists in a short time; thus, it is crucial to respond to such queries quickly using as few as possible of peers.

In distributed systems, traditional skyline algorithms do not consider keywords in their design. The work in [5] [6] [7], for example, considers values in dimensions to distribute tuples. These studies exploit prune-ability and incomparable partitions of skyline queries. Nevertheless, they do not consider keywords. Modifying them to satisfy users' requirements by ignoring undesired discovered points results in traversing unneeded peers. On the other hand, traditional keyword search query [8] and Napster [9] ignore skyline incomparability and prune-ability features. To exploit skyline pruning ability and incomparability as well as keyword search algorithms, this paper devise the keyword-matched skyline algorithms to combine keyword search and skyline algorithms to efficiently answer keyword-matched skyline. The algorithms keep some order of the peers to exploit prune-ability and incomparability. They also use DHT functions, Bloom filters and Cover-set features to keep track of points' and peers' keywords as explained in Section 3.

The contributions of this paper are as follows:

- Bloom filters are used to figure out the candidate peers for query keywords with cover-set tuples and nodes.
- Keyword-matched skyline algorithms are designed and implemented in P2P systems.
- Experiments have been carried out and show that the proposed approaches resulted in reduction of traversed peers while preserving progressiveness.

The rest of the paper is organized as follows. We first discuss related work in Section 2. Problem definition and algorithms are discussed in Section 3. Section 4 discusses experiments and our findings. We conclude our paper in Section 5.

2 Related Work

Börzsönyi's paper [1] was the first work to introduce skyline into databases. Block Nested Loop (BNL) uses a window to compare all points and discover the skyline points. Nearest neighbor (NN) [10] used R-trees and a *to-do* list to get skyline; branch and bound skyline (BBS) [11], however, uses R-trees and a heap to get rid of duplicates introduced in NN.

Because BNL, BBS, NN and the other centralized algorithms are not efficient for distributed and P2P systems, the distributed algorithms have been suggested. In [5], for example, data are distributed vertically. The traditional skyline is retrieved using a round-robin on the presorted attributes. In the feedback-based distributed skyline algorithm (FDS) [12], the coordinator iteratively contacts the other nodes providing a feed-back. Some algorithms [6][13] have converted multi-dimensional data into a single-data index and adapted it into P2P.

Other types of skyline queries (e.g. Subspace skyline, constrained skyline queries) in P2P systems have also been considered in literatures. For constrained skyline, data space is partitioned horizontally in the Distributed SkyLine query

(DSL) [14]. SkyFrame [7] uses greedy and relaxed skyline search on Baton (a balanced tree structure for peer-to-peer networks). In the Parallel Distributed Skyline (PaDSkyline) [15] and SkyPlan [16], the querying peer collects the minimum bounding rectangles (MBRs) from other peers, and with different measures (e.g. weighted edges and spanning trees), a plan is mapped for incomparable peers to work in parallel.

Skypeer [17] and Distributed Caching Mechanism (DCM) [18] are meant for subspaces skyline queries in distributed systems. A super-peer architecture is used for Skypeer. They defined the *extended skylines* which are collected by super-peers from the other peers. Queries are submitted to a super-peer which contacts the other super-peers for their subspace skyline. DCM explores caching. The results of subspaces queries are cached in peers using a distributed cache index (DCI) on Baton or Chord. The subspace queries use the DCI to forward next subspace skyline queries. Hose and Vlachou [19] have studied skyline processing in distributed systems in more details.

Even though the above algorithms answer skyline query efficiently, they are not designed to consider keyword-matched skyline. They all only use quantity values in the attributes to take advantage of skyline feature of prune-ability and incomparability. Attributes that may be boolean is not supported. A modification of those algorithms to satisfy user's requirements can reflect inefficiency.

Keyword-matched skyline has been introduced in [20]. It uses an R-tree. While R-trees are efficient for centralized systems, they are inefficient for P2P systems due to the heavy data volumes required for maintenance. In this paper, we investigate the keyword-matched data skyline in P2P systems using Chord [3] as our overlay. Other overlay structures may also be applied.

For keyword query, the traditional techniques either used centralized search as Napster [9], query broadcasting as Gnutella [21], or well-known naming such as Freenet [22]. They are not efficient for skyline queries because they do not exploit skyline properties, such as pruning ability and incomparability.

To the best of our knowledge, keyword-matched skyline has not been considered in distributed and P2P systems. Our aim is to minimize the visited peers in the network while preserving progressiveness.

3 Keyword-Matched Skyline Queries in P2P Systems

Some formalizations to keyword-matched skyline are presented in Section 3.1. Algorithms for keyword-matched skyline in P2P systems will be discussed in the next three Sections.

3.1 Problem Definition

In this subsection, some definitions are stated for keyword-matched skylines. Without loss of generality, we assume minimum values of attributes are preferred (e.g. cheaper is preferred to expensive, less mileage is preferred to high mileage, etc). For maximum value preferences, the inverse of the values can be used.

A tuple t in a d-dimensional space D^d is defined as <V,W> where V = $(v_1, v_2, ..., v_d)$ is a value vector of d-numerical values; and W = $(w_1, w_2, ..., w_k)$ is a set of k keywords for the tuple t. In addition, the value vector of a tuple t_i is denoted by $t_i.V$, while its set of keywords is denoted by $t_i.W$.

Definition 1. *A keyword-matched tuple to a query keyword ($Q_k(D^d, W)$). For a query q with a set of query keywords $q.W$, a tuple t is a keyword-match tuple to the query q if and only if $\forall w \in q.W$, $w \in t.W$.*

Definition 2. *Domination. Let t and t' be two tuples in D^d, where $t.V = (v_1, v_2, ..., v_d)$ and $t'.V = (u_1, u_2, ..., u_d)$. Then, t dominates t' ($t \prec t'$) if and only if $\forall i$, $v_i \leq u_i$ and $\exists i$, $v_i < u_i$. Conversely, t does not dominate t', denoted $t \not\prec t'$ if and if only $\exists i, v_i > u_i$.*

Definition 3. *Skyline Tuple. In skyline operator ($Q_s(D^d)$), a tuple t in D^d is a skyline tuple if and only if $\nexists t' \in D^d$; $t' \prec t$*

Definition 4. *A Keyword-matched Skyline Tuple. Let A be all keyword-matched tuples to a query q with keywords W. A tuple $t \in A$ is a keyword-matched skyline tuple to the query q if and only if $\forall t' \in A$; $\nexists t' \prec t$.*

Definition 5. *A keyword-matched skyline query ($Q_{ks}(D^d, W)$). Given a set of query keywords W and a dataset D^d, a keyword-matched skyline query denoted as $Q_{ks}(D^d, W)$, retrieves the set of skyline tuples whose each textual attribute contains all words of W. Thus, the following equivalent rule is true:*

$$Q_{ks}(D^d, W) \equiv Q_s(Q_k(D^d, W)) \tag{1}$$

Definition 6. *Let m be $min_x \in D(x_{min})$, Initial Skyline Peers P and Candidate Skyline Points M as*
$P = \{P_i | \exists x \in D_{P_i} \text{ such that } x_{min} = m\}$
$M = \{x | x \in D_P \wedge x_{min} = m\}$,
where $D_P = \bigcup_{P_i \in P} D_{P_i}$.

Theorem 1. *If S_D and S_M are the skylines of D and M respectively, then $S_M \subseteq S_D$ and $S_M \neq \phi$.*

Proof. From Definition 6 for M, for any $x \in S_M$, x is not dominated by any other point in M. Suppose $m' = min(x'_{min})\ \forall\ x' \in D - M$. As m < m', x can not be dominated by any $x' \in D - M$. Therefore, x is not dominated by any other point in D, i.e., $x \in S_D$. Thus, we have $S_M \subseteq S_D$. Since M is not empty, hence $S_M \neq \phi$.

Theorem 2. *A tuple t_1 with a minimum value $t_1.v_i$ in a dimension i can not be dominated by any point t_2 with a minimum value $t_2.v_j$ in any dimension j where $t_1.v_i < t_2.v_j$.*

Proof. Let's assume t_2 dominates t_1. Thus, $t_2.v_i \leq t_1.v_i$. Since $t_2.v_i \leq t_1.v_i \Rightarrow t_2.v_i < t_2.v_j$. This contradicts that $t_2.v_j$ is the minimum value of t_2. If $t_2.v_i$ is the minimum value of y, the condition of our theorem is not satisfied. On the other hand, if $t_2.v_i > t_1.v_i$, t_2 does not dominate t_1.

In the next three sections, we present three algorithms: a baseline (Ch-isky) which is based on a state of the art, and two new algorithms: Node-based keyword skyline (NK-sky) and Cover-based keyword skyline (Ck-sky). Definition 6 and Theorem 1 of isky[6] [13] are modified here to fit to our problem. Note that nodes and peers are used interchangeably to mean peers in this paper.

3.2 Chord-Based isky (Ch-isky)

The baseline approach is based on isky [6] [13]. It is applied on Chord overlay which we call Ch-isky. The algorithm is shown in Algorithm 1. As in isky, data values in each dimension are assumed to be in the period [0, 1]. The period [1, d+1] is distributed among peers with an equal continuous periods. Each peer is responsible of the next period in clockwise fashion starting with peer 0. A tuple is looked for its minimum value in all of its dimensions. The sum of the minimum value and the dimension it is found in is used to determine its destination peer when it is distributed. If the minimum value is found in more than one dimension, the lowest dimension is taken.

Once a query is triggered, it blindly travels through nodes exploiting pruneability. Lines [3-7] in Algorithm 1 require the querying peer to first broadcast the query to all Initial Skyline peers P (i.e. all peers that include the dimension values {1,2, ... ,d} in their period). In line 8, the skyline tuples are calculated using the volume filter and the min-max pruning ability to reduce calculations. In line 9, a new volume filter and a new min-max values are found using Equations 3 and 4. Line 10, the keyword-matched skyline tuples are sent to the querying peer and then returned to the user if these tuples exactly form a skyline. A volume filter and the min-max pruning are used. In lines [11-13], the query travels from a peer to the next peer in a clockwise fashion.

Our next two algorithms are based on the DHTs for the keywords and Bloom filters to minimize the number of candidate peers for keyword-matched skyline query. Even though Bloom filters can introduce few false positives, they do not affect the correctness of our algorithms as shown later.

Bloom Filters. To summarize membership in a set, a hash-based data structure called a Bloom filter [8] is used. A peer A can send its Bloom filter for its elements set E_A to another peer B with an element set E_B instead of sending its elements set. Thus, it reduces the amount of communication required for a peer to determine $A \cap B$. The membership test will never return false negatives, but it may return false positives with a tunable, predictable probability as shown in Equation 2 [8]. The results of the intersection in a peer E_B with E_A will contain all of the true intersection and may have also a few (false positive) hits that are only in E_B and not E_A. As the size of the Bloom filter increases, the number of false positives falls exponentially. Equation 2 predicts the probability of a false positive p_{fp} when an optimal choice of hash functions is given, and the Bloom filter bits m, and the number of elements in the set n are also given.

Algorithm 1. Ch-isky: Chord-based isky algorithm

```
1: Input: MinMax-filter, VDR, Querying-Peer, keywords
2: BEGIN
3: if QueryingPeer then
4:    for each Peer P includes an integer (1 to D) in its period do
5:       send Keyword-matched-Skyline-Query
6:    end for
7: end if
8: Calc-key-matched-sky-using-VDR-MinMax()
9: Calc-VDR-and-MinMax() /* using Equations 3, 4 */
10: Send-results-to-query-peer
11: if min(nextPeer) ≯ MinMax then
12:    send-query-to-next-peer(MinMax,VDR)
13: end if
14: END
```

Thus, to maintain a low false-positives probability, the Bloom filter size needs to be proportional to the number of represented elements.

$$p_{fp} = 0.6185^{m/n} \tag{2}$$

3.3 Node-Based Keyword-Matched Skyline Algorithm (Nk-sky)

In this section, we propose Nk-sky, a node-based keyword-matched skyline algorithm. For construction of Nk-sky, tuples are distributed to the peers according to the sum of their minimum values and dimension of the minimum value as explained above in Ch-isky. Each peer builds its keyword set. The peer's keyword set is the union of all points' keywords in a peer. It hashes each of its keywords using the DHT functions and sends them along with the peer's id to the keyword responsible peers. The keyword responsible peer hashes the node ID into the Bloom filter of that keyword.

The Nk-sky skyline query runs through two stages: 1) discovering candidate peers, and 2) skyline calculation.

1) Discovering Candidate Peers. Once the query is triggered, the querying peer hashes one keyword using the DHT function and sends the query to the responsible peer. The responsible peer gets the query peer Bloom filter for that query keyword and sends it to the next keyword's responsible peer. Each responsible peer receiving a Bloom filter would do the intersection with its keyword's Bloom filter and sends the results to the next responsible peer until all query keywords are processed. The last peer sends the results of the intersections to the querying peer. The results are the nodes with all query keywords. There might be few false positives but they will not affect correctness of the query results. For each keyword, it may require at most O(log n) lookup messages. Thus, for k query keywords, at most O($k$log n) lookup messages may be expected.

2) Skyline Calculations. At this stage, candidate peers, in addition to a few false positives, are known. The false positives will only affect traversing those peers unnecessarily but they will not affect the correctness of the results. The skyline query chooses peers from the candidates to broadcast to. A peer is chosen if it is the closest above or equal candidate peer to any peer responsible of the periods that include {1,2, . . .,d}. A peer c is *the closest above or equal candidate to another peer responsible of a value* i if and only if there is no other peer in the set that has a value v closer to i than any value in c and both v and c greater or equal to i. For example, suppose a peer p is responsible of period [1.8, 2.2) if p is in candidate peers, the query is sent to it because it includes 2. If p is not in the candidate peers, the query is sent to the closest candidate peer within the period [2.2, 3). Because there are at most d peers that include i ∈ {1,2, ... ,d}, the query broadcasts to a maximum of d candidate peers . All query processing is done in parallel.

In Nk-sky, a peer processes the keyword-matched skyline query and sends its results to the querying peer. The querying peer returns the results to the user if no future point can dominate them progressively. The processing peer also sends the query, a max-min value filter [6][13] and a Volume of Dominating Region (VDR) [23] to the closest above candidate peer in a clockwise fashion.

$$SF_{global} = min_{x \in S}(x_{max}) \tag{3}$$

In Equation 3, the min-max-value filter (SF_{global}) is used because we use minimum value as opposed to maximum value used in [6][13]. It is obtained from the already found skyline points (S). A peer is pruned if its minimum value is greater than SF_{global}. VDR, shown in Equation 4, is used to prune points within a peer. It is expected to prune more points than others due to its volume.

$$VDR_p = \prod_{i=1}^{d}(b_i - p_i) \tag{4}$$

where b_i is the maximum value in dimension i, and p_i is the value of p in dimension i. A peer and the following peers can be pruned if its minimum value is greater than SF_{global}.

Lemma 1. *Keyword-matched skyline results of Nk-sky are correct and complete.*

Proof. Correctness. Let t_1 and t_2 be two tuples with v_1^i and v_2^i be values in dimension i, respectively. Let t_1 dominates ($\prec$) t_2. If t_1 and t_2 have their minimum values in dimension i (i.e. v_1^i and v_2^i, respectively), t_1 is visited before t_2. Thus, t_2 will be pruned. On the other hand, lets t_1 and t_2 have their minimum values in different dimensions (i.e. v_1^i and v_2^j, respectively). From Theorem 2, in the algorithm, t_2 will not be declared as a keyword-matched skyline tuple until t_1 is seen. t_1 will prune t_2. Thus, no false skyline tuple will be produced.

Completeness. All tuples are checked. A peer is pruned if its minimum values are greater than the maximum value of a skyline tuple found so far. No tuple is pruned if not dominated.

Algorithm 2. Ck-sky: cover-based keyword-matched skyline algorithm

```
 1: Input: MinMax-filter, VDR, QueryingPeer, keywords, Peers-to-be-traversed
 2: BEGIN
 3: if QueryingPeer then
 4:    /* first stage */
 5:    Peers-to-be-traversed = get-candidate-peers-using-BloomFilter()
 6:    for all P ∈ Peers-to-be-traversed do
 7:       if p closest above or equal peer to an integer (1 to D)  then
 8:          send keyword-matched-skyline-query
 9:       end if
10:    end for
11: end if
12: /* Second stage */
13: Calc-keyword-matched-skyline-using-VDR-MinMax()
14: Calc-VDR-and-MinMax() /* using Equations 3, 4 */
15: Send-results-to-query-peer
16: nextPeer = closest-candidate-peer-in-an-increase-order
17: if  min(nextPeer) ⊁ MinMax then
18:    send-query-to-next-peer(MinMax,VDR)
19: end if
20: END
```

3.4 Cover-Based Keyword-Matched Skyline Algorithm (Ck-sky)

Using nodes instead of tuples in Nk-sky in Section 3.3 seems to be natural and more attractable to reduce false positives according to Equation 2. However, in reality, it is not the case for two reasons as shown in the experiments:

1) Node's keywords produced by OR-ing do not mean a node have a tuple with all query keywords.

2) Skyline query algorithms use pruning ability, which reduces the peers with query keywords as well as false positives from the candidates.

We propose Ck-sky, a cover-based keyword-matched skyline algorithm, as shown in Algorithm 2. In Ck-sky, a keyword-matched skyline query also runs in the same two stages of Nk-sky. However, we use tuple keywords instead of node keywords in Ck-sky. In lines [3-11], the first stage is presented. Thus, instead of sending only a node id to the keyword responsible peer, the tuple id is also sent. Bloom filter is used for the tuple ids. In the first stage, the querying peer gets the candidate peers using bloom filter as shown in line 3. In lines [6-10], instead of going to the dimension values peers discussed in Ch-isky, only the candidate peers closest to the dimension values peers are used to start the query. Lines [12-19]are responsible of the second stage. Line 13 calculates the skyline using the VDR and MinMax filters. The new VDR and MinMax are calculated in line 14 using Equations 3 and 4. The results are sent to the querying peer in line 15. The next peer to visit is calculated in line 16. In line 18, the query is sent to next peer if it is not pruned.

To reduce the number of tuples sent and maintained by the Bloom filter, we suggest Cover set.

Cover Tuple: A tuple t is said to be a cover for a tuple t' if and only if $\forall w \in t'.W$, w$\in t.W$.

Mutual Cover Tuple Set: All tuples that cover each other in set (i.e. they have the same keywords).

Cover Set: A cover set is the set of all tuples that have no cover in a node in addition to a tuple from each mutual cover tuple set.

In Ck-sky, a peer can send its Cover set instead of sending all tuples at the distribution phase. In the first stage, tuples are considered in the Bloom filter as opposed to nodes in Nk-sky. The second stage are done as in Nk-sky.

In the first stage, a query may need O($k$log n) lookup messages to reach to all k query keyword peers for their Bloom filters. It is, however, optimized by visiting keywords peers in order (in a clockwise order).

In the second stage, for an m candidate peers, a maximum of m jumps may be needed. Each jump may require O(log n) lookup messages. Due to the traversal order of our algorithms, the larger distance between candidate peers, the larger number of peers pruned. The portion of the m candidate peers that may be pruned, however, depends on the tuples in each peer.

Lemma 1 is also applicable for both Nk-sky and Ck-sky. The following Lemma is complement to Lemma 1.

Lemma 2. *Covered set does not affect the correctness of the keyword-matched skyline query.*

Proof. Let's assume a peer p with a tuple t. t is either in the Cover set of p or not. If t is in Cover set, then p will be included in the candidate peers. Let's assume t is not in the Cover set of p. This means there is t' that covers t if t is keyword-matched tuple to a query keyword q (Definition 1). Since ($\forall w \in q.W \rightarrow w \in t.W$) and ($\forall w' \in t.W \rightarrow w' \in t'.W$) $\Longrightarrow$ ($\forall w'' \in q.W \rightarrow w'' \in t'.W$). Thus, the peer p will be included in the candidate peers.

4 Performance Evaluation

In this section we evaluate our algorithms by checking the reduction of the visited peers for keyword-matched skyline queries. For a thorough investigation, we use synthetic datasets in our experiments to show the reduction in traversed peers for different variances (parameters). Table 1 summarizes the used parameter settings. The parameters include value distribution, dimensionality, cardinality, query keywords size, network size, and skew factor of the word distribution.

As in [1], for the experiments, we generated three types of synthetic datasets: (i) independent datasets, (ii) correlated datasets and (iii) anti-correlated datasets with 1000 distinct keywords.

Table 1. Parameter settings in the experiments

Parameters	Values
Cardinality(N) of tuples	100k, 200k, 400k, 600k, 800k, **1M**
Dimensionality	2, **3**, 4, 5
The number of query words (k)	1, **2**, 3, 4, 5
Zipf skew factor (θ)	0.0, 0.2, 0.4, 0.6, **0.8**, 1.0
Distribution(for values)	independent, correlated, anti-correlated
Tuple's keywords	6
Network size (no. of peers)	100, 1000, **2000**, 3000, 4000

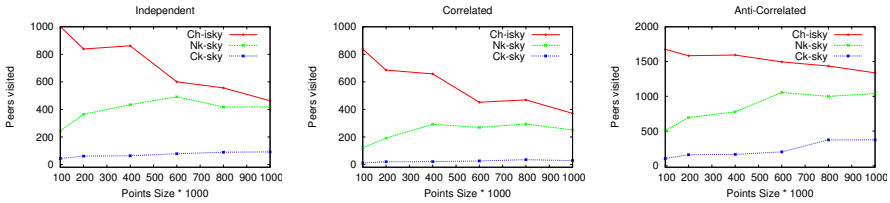

Fig. 2. Visited Peers vs. Number of Tuples

The three types of datasets are usually used for the evaluation of skyline. In the independent datasets, the values in each dimension is independent of each other; while in correlated datasets, the values are correlated as in a good grades of one student may come with more papers published. On the other hand, in anti-correlated datasets, a better value in one dimension will probably mean a worse value in the other dimensions as in the hotel example where a closer distance hotel to the beach will be a more expensive hotel.

The experiments were carried out on Intel(R) Core(TM) i3 CPU M350 (2.27 GHz), 3 GB RAM using Peersim-1.0.5 [24].

Our Ck-sky has shown to perform better than the other two algorithms in reducing the visited peers for a query.

4.1 The Effects of Cardinality

In this section, we show our findings with experiments to evaluate scalability with respect to dataset cardinality. Our experiments were done for various cardinalities with the range [100k,1M] and the default parameters shown in Table 1. Fig. 2 depicts our findings.

It shows that in all distribution of values (independent, correlated, and anti-correlated), Ck-sky preforms better. These results come from the fact that, in Ck-sky, visited peers only will probably contribute to the answer to the query. Using the minimum values to distribute tuples also contribute to minimizing the traversed peers by pruning peers that definitely can not contribute to the answer.

Nk-sky performs better than Ch-isky because in Nk-sky, only suspected peers are visited. Ch-isky, however, visits all peers in order of minimum values until it finds a pruning answer. In the independent and anti-correlated distributions,

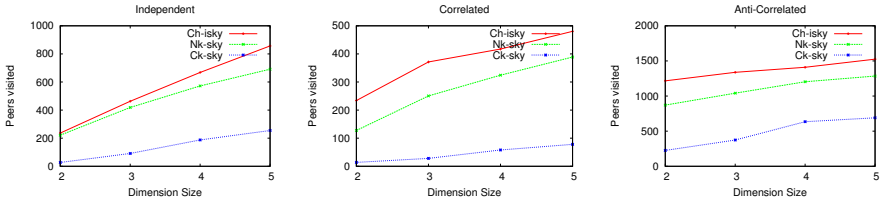

Fig. 3. Visited Peers vs. Dimensions

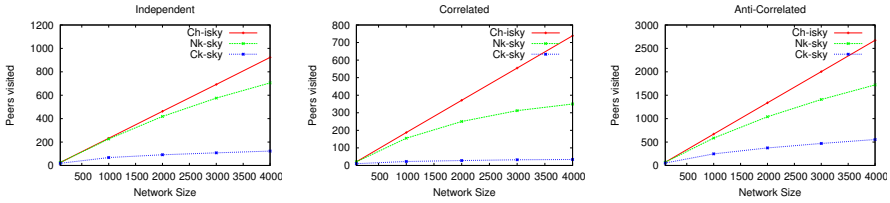

Fig. 4. Traversed Peers vs. Network Size

the curves for the Ch-isky and Nk-sky algorithms are higher than the curves in correlated. In correlated datasets, fewer skyline points are expected due to their correlations; and they can be found in early peers.

4.2 The Effects of Dimensionality

As the number of dimensions (attributes) increases, an increase in skyline points is expected. Larger variation among tuples is expected when higher dimensions are used. In this section, we show the effect of dimensions on the number of traversed peers in our algorithms. Our experiments are done with different dimensions [1D,2D ... 5D] with default values shown in Table 1. Fig. 3 goes along with our expectations that as dimensions increase, the number of traversed peers increases. Ck-sky is still better than the other algorithms. Ck-sky's increase as the dimensions increase is also expected as the pruning ability decreases with higher dimensions. In anti-correlated distribution, visited peers for all algorithms show a slight increase in traversed peers because the expected skyline points are also higher. All algorithms do better in correlated database distribution than the others because the correlation between values results in a better pruning. As stated earlier, Ck-sky visits only peers that will probably have skyline points.

4.3 The Effects of Network Size

As network size increases, we expect traversed peers to increase. How does the increase vary with our algorithms? In this section, we answer this question. Fig. 4 shows that the increase in Ch-isky is with a slope of one. The OR-ing used in the Nk-sky algorithm has more effects as the number of tuples in a peer decreases due to the increase of network size. Nk-sky becomes better than Ch-isky depending on the network size and type of data set. However, the affect is not as good as the

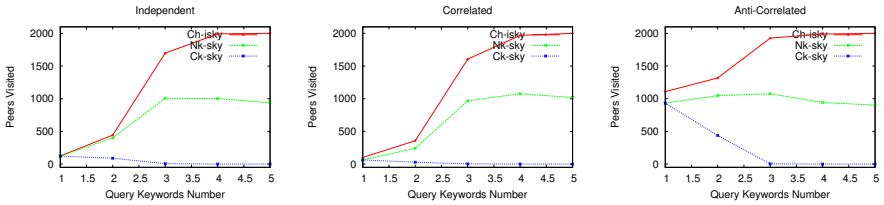

Fig. 5. Traversed Peers vs. Query Keyword Size

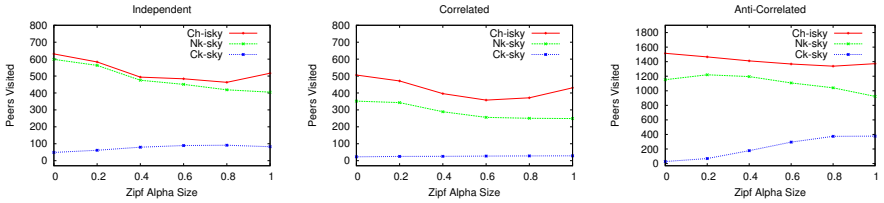

Fig. 6. Traversed Peers vs. Skewness

Ck-sky algorithm. Ck-sky traversed peers increase is expected as skyline tuples are distributed to more peers when network size increases. Thus, the increase in traversed peers in Ck-sky for the three types of data sets complies with [1]. Skyline tuples are more for the anti-correlated than the other models. Traversed peers increase in this model with our experiments for the same reason. Its effect is less for the other models due to the wrong peers traversed.

4.4 The Effects on the Size of Query Keywords

Query keywords can also affect traversed peers. As query keywords increase, fewer tuples would probably satisfy the query. This is also shown in Fig. 5 for Ck-sky. Due to the false peers that result from the other algorithms, more peers are visited. For Ck-sky and Nk-sky they will have the same number of visited peers when the query keyword is one because Nk-sky will not have false peers. False peers increase as we go farther. Ch-isky can probably have the same traversed peers at the beginning due to the probability that a peer will have tuples with fewer keywords than many keywords. The difference becomes big as the system does not have a tuple with the query keywords. However, Ck-sky can discover this at the first stage.

4.5 The Effects of Word Distribution

The results in Fig. 6 might be surprising because more traversed peers are expected in Ch-isky and Nk-sky because more tuples will satisfy the query. The reduction should not be surprising for two reasons: 1) Even though more peers are supposed to have satisfied tuples, they are contained in the unnecessary peers that are traversed with low zipf factor. 2) The skyline algorithms exploit pruning. Thus, because the probability of visiting a real candidate peer increases and

the prune-ability increases for the wrong peers. However, this reduction is not big due to the big probability of a wrong visited peer. The increase in traversed peers in Ck-sky is due to more tuples are expected to be in the solution. This is shown more obviously in anti-correlated distribution where more tuples are expected. For independent distribution, the number of traversed peers is slightly higher than correlated distribution due to the prune-ability and the results size.

5 Conclusion

This paper addresses keyword-matched skyline in peer-to-peer systems. Tuples may have keywords in addition to value (quantity) attributes. Keywords are boolean attributes that a tuple may have or may not. The designs of the traditional skyline algorithms only consider value attributes in all tuples. By specifying some keywords in the query, a user needs a skyline for only tuples with these keywords. It is called keyword-matched skyline. Modifying traditional skyline algorithms is inefficient. The traditional keyword algorithms are not good for keyword-matched skyline because they do not exploit prune-ability found in skyline queries.

In this paper, node and tuple-based algorithms are designed to solve keyword-matched skyline in peer-to-peer systems efficiently. The algorithms use DHTs functions and Bloom filters to minimize the number of traversed peers. Cover sets are also defined for peer's tuples to reduce false positives peers resulted from Bloom filters. Results show that tuple-based cover set (Ck-sky) algorithm performs better than the other algorithms. It considers only necessary tuples (cover set) in a node when keywords are hashed. Even though we studied the keyword-matched skyline in P2P systems in this paper, other issues could also be investigated, such as keyword-matched skyline in streams, subspaces and probability of keyword-matched skyline. Those issues are to be considered for the future work.

Acknowledgments. This work is supported by National Natural Science Foundation of China under Grants 61173170 and 70771043, National High Technology Research and Development Program of China under Grant 2007AA01Z403, and Innovation Fund of Huazhong University of Science and Technology under Grants 2012TS052 and 2012TS053.

References

1. Börzsönyi, S., Kossmann, D., Stocker, K.: The skyline operator. In: The 17th International Conference on Data Engineering (ICDE 2001), pp. 421–432 (2001)
2. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S.: A scalable content-addressable network. In: ACM SIGCOMM, pp. 161–172 (2001)
3. Stoica, I., Morris, R., Karger, D.R., Kaashoek, M.F., Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for internet applications. In: SIGCOMM, pp. 149–160 (2001)
4. Zhuge, H., Chen, X., Sun, X., Yao, E.: Hring: a structured p2p overlay based on harmonic series. IEEE Transactions on Parallel and Distributed Systems 19(2), 145–158 (2008)

5. Balke, W.-T., Güntzer, U., Zheng, J.X.: Efficient distributed skylining for web information systems. In: Bertino, E., Christodoulakis, S., Plexousakis, D., Christophides, V., Koubarakis, M., Böhm, K. (eds.) EDBT 2004. LNCS, vol. 2992, pp. 256–273. Springer, Heidelberg (2004)
6. Cui, B., Chen, L., Xu, L., Lu, H., Song, G., Xu, Q.: Efficient skyline computation in structured peer-to-peer systems. IEEE Transactions on Knowledge and Data Enginering 21(7), 1059–1072 (2009)
7. Wang, S., Vu, Q.H., Ooi, B.C., Tung, A.K.H., Xu, L.: Skyframe: a framework for skyline query processing in peer-to-peer systems. VLDB Journal 18(1), 345–362 (2009)
8. Mullin, J.: Optimal semijoins for distributed database systems. IEEE Transactions on Software Engineering 16(5), 558–560 (1990)
9. http://www.napster.com/
10. Kossmann, D., Ramsak, F., Rost, S.: Shooting stars in the sky: An online algorithm for skyline queries. In: VLDB, pp. 275–286 (2002)
11. Papadias, D., Tao, Y., Fu, G., Seeger, B.: Progressive skyline computation in database systems. ACM Transactions on Database Systems 30(1), 41–82 (2005)
12. Zhu, L., Tao, Y., Zhou, S.: Distributed skyline retrieval with low bandwidth consumption. IEEE Transactions on Knowledge and Data Enginering 21(3), 384–400 (2009)
13. Chen, L., Cui, B., Lu, H., Xu, L., Xu, Q.: isky: Efficient and progressive skyline computing in a structured p2p network. In: The 28th International Conference on Distributed Computing Systems (ICDCS 2008), pp. 160–167 (2008)
14. Wu, P., Zhang, C., Feng, Y., Zhao, B.Y., Agrawal, D., El Abbadi, A.: Parallelizing skyline queries for scalable distribution. In: Ioannidis, Y., Scholl, M.H., Schmidt, J.W., Matthes, F., Hatzopoulos, M., Böhm, K., Kemper, A., Grust, T., Böhm, C. (eds.) EDBT 2006. LNCS, vol. 3896, pp. 112–130. Springer, Heidelberg (2006)
15. Chen, L., Cui, B., Lu, H.: Constrained skyline query processing against distributed data sites. IEEE Transactions on Knowledge and Data Enginering 23(2), 204–217 (2011)
16. Rocha-Junior, J., Vlachou, A., Doulkeridis, C., Nørvåg, K.: Efficient execution plans for distributed skyline query processing. In: EDBT, pp. 271–282 (2011)
17. Vlachou, A., Doulkeridis, C., Kotidis, Y., Vazirgiannis, M.: SKYPEER: Efficient subspace skyline computation over distributed data. In: ICDE, pp. 416–425 (2007)
18. Chen, L., Cui, B., Xu, L., Shen, H.T.: Distributed cache indexing for efficient subspace skyline computation in P2P networks. In: Kitagawa, H., Ishikawa, Y., Li, Q., Watanabe, C. (eds.) DASFAA 2010, Part I. LNCS, vol. 5981, pp. 3–18. Springer, Heidelberg (2010)
19. Hose, K., Vlachou, A.: A survey of skyline processing in highly distributed environments. The VLDB Journal–The International Journal on Very Large Data Bases 21(3), 359–384 (2012)
20. Choi, H., Jung, H., Lee, K., Chung, Y.: Skyline queries on keyword-matched data. Information Sciences (2012), doi:10.1016/j.ins.2012.01.045
21. http://rfc-gnutella.sourceforge.net/developer/stable/index.html
22. Clarke, I., Sandberg, O., Wiley, B., Hong, T.W.: Freenet: A distributed anonymous information storage and retrieval system. In: Federrath, H. (ed.) Anonymity 2000. LNCS, vol. 2009, pp. 46–66. Springer, Heidelberg (2001)
23. Huang, Z., Jensen, C., Lu, H., Ooi, B.: Skyline queries against mobile lightweight devices in manets. In: Proceedings of the 22nd International Conference on Data Engineering (ICDE 2006), pp. 66–66 (2006)
24. http://peersim.sourceforge.net/