

# Efficient Progressive Processing of Skyline Queries in Peer-to-Peer Systems

Huajing Li Qingzhao Tan Wang-Chien Lee

Department of Computer Science and Engineering

Pennsylvania State University

University Park, PA 16802, USA

Email: {huali, qtan, wlee}@cse.psu.edu

**Abstract**—Skyline queries have received a lot of attention from database and information retrieval research communities. A skyline query returns a set of data objects that is not dominated by any other data objects in a given dataset. However, most of existing studies focus on skyline query processing in centralized systems. Only recently, skyline queries are considered in a distributed computing environment. Acknowledging the trend toward peer-to-peer (P2P) systems in distributed computing, we examine the problem of skyline query processing in P2P systems and propose innovative solutions. We exploit the data semantic embedded in semantically structured P2P overlay networks to efficiently prune search space, without compromising the quality of query result. In addition, we propose approximate algorithms to support skyline queries where exact answers are too costly to obtain. These approximate algorithms produce high quality answers using heuristics based on local semantics of peer nodes. Extensive experiments validate that our algorithms provides high efficiency and scalability to skyline query processing in P2P systems.

## I. INTRODUCTION

Peer-to-peer (P2P) systems, widely used for information sharing and exchange amongst millions of users, is a new paradigm for distributed computing. In contrast to the traditional client-server model, peer nodes in the P2P paradigm act both as an information provider (server) and a consumer (client) at the same time. Additionally, P2P systems have the following advantages:

- P2P systems usually do not place strict requirements on capacities of participating peers.
- P2P systems avoid single-point-of-failure, which makes the system more robust.
- P2P systems are self-adaptive, significantly reducing the system maintenance cost.
- P2P systems publish up-to-date information to users in a timely fashion.

While the traditional client-server based distributed systems will stay, there is a trend towards supporting existing and new services and functionalities in the alternative P2P paradigm. Algorithms have been proposed to support various query types, such as range queries [5],  $k$  nearest neighbors queries [11], and top- $k$  queries [4], in P2P systems.

Skyline has arisen as an important query type in databases and information systems. A skyline query returns a set of data

objects that is not *dominated* by any other data object in a given dataset. Here, dominance among objects are determined by search criteria. An object dominates another one when it is not worse in all search criteria and is better in at least one criterion. The classical example of skyline queries is a multi-criteria decision making problem: *finding a hotel with nearer distance to a beach and a lower price*. Usually the hotel near a beach will be more expensive, which makes it almost impossible to find a hotel with both the minimal distance and lowest price. Therefore, the skyline query would return a set of non-dominated hotels for a user to choose, with all dominated hotel records eliminated.

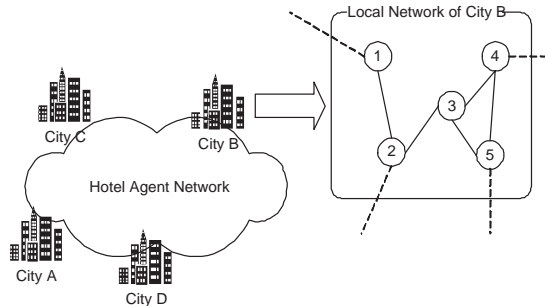


Fig. 1. Example of a P2P Hotel Agent Network.

Skyline queries are particularly useful for the users exploring what is available in the system. Taking an Internet-scale hotel management system as an example, hotel information, such as reservation status and room prices, may change over time. Thus, a P2P network consisting of hotel agencies which provide up-to-minute hotel information is desirable. In such systems, each agency is only responsible for its local hotel information. Users can connect to any agency (i.e., peer) to search for hotel rooms of their preference. In Figure 1, City B has five registered hotel agents, each of which has some connections to peer agents either in the same city (represented with solid lines) or in other cities (represented with dotted lines). Suppose a user wants to reserve a double room with a preference of lower price and nearer distance to the beach in City B. He may issue a skyline query as follows (in a SQL-like

syntax extended with preference):

```
SELECT name
FROM hotels
WHERE city='B'
AND available-doubleroom>0
PREFERRING doubleroom-price LOW,
dis-beach LOW
```

Skyline problem has been well studied in centralized database environments (see [13] for a comprehensive survey). However, none of the existing algorithms are applicable in a highly dynamic P2P system. Building and maintaining a specialized index for skyline queries (e.g., [18]) may be futile and unrealistic for P2P systems due to the huge network size and limited resources of individual peers. In addition, a peer can not be always available, making previous query results out-of-date. In this paper we propose P2P skyline algorithms, exploiting the nature of semantically structured overlay networks to effectively restrict the network scope that is involved in the query execution without affecting the accuracy of results. A semantic overlay network, Semantic Small World (SSW), is chosen as the testbed of our algorithm.

The primary contributions of this work are three-fold:

- We propose an *exact* progressive skyline algorithm for P2P overlay networks, which scales well with network size and data dimensionality.
- For P2P systems that are not semantically structured, we propose *approximate* algorithms that produce good quality skylines without incurring excessive execution costs.
- We define the evaluation metrics in measuring the efficiency of P2P skyline algorithms and perform an extensive, simulation-based, performance evaluation.

The rest of this paper is organized as follows. In Section II, we review some related works in skyline algorithms and P2P networks. An exact P2P skyline algorithm is proposed and discussed in Section III. In Section IV, we develop efficient approximate algorithms for P2P systems where the exact algorithm cannot be employed. The simulation setup and evaluation results are presented in Section V. Finally, we give our concluding remarks and discuss future study plans in Section VI.

## II. PRELIMINARIES AND RELATED WORK

### A. Overview of P2P Systems

In contrast to the traditional client-server computing model, a peer in a P2P system can act as both a server as well as a client. P2P systems need to be scalable to the massive amount of dynamically changing data. How to efficiently search for information in P2P systems is a crucial research issue. Gnutella [1] floods the network to search the desired information, Random Walk [12] traverses the P2P network through a number of randomly selected paths in the network. *Centralized index*, such as Napster [2], stores the index at one central server, which makes a P2P system act like a

server-client system. *Distributed index* P2P system distributes the index table amongst peers in the network. Some index structures keep track of the location information of the peers, such as Local Index [20] and Neighborhood Signature [10]. Others also maintain the key proximity of each peer by letting similar peers connected to each other. They build a structured overlay on top of the peers. Examples of such P2P systems include CAN[15], CHORD[17], TAPESTRY[22], and PASTRY[16].

The *Semantics based* P2P networks take into account the content of the data objects, e.g., pSearch [19] and SSW [9]. Our previous work, SSW [9], builds an overlay network and distributed index structure for semantic based P2P search. This mechanism dynamically clusters peers with semantically similar data closer to each other and then maintains the clusters in a semantic index structure. Experiments show that such a cluster-based P2P overlay structure improves the search efficiency greatly.

### B. Skyline Algorithms

During the last couple of years, a series of skyline algorithms have been proposed. *Block Nested Loop (BNL)* [6] compares each point  $p$  with every other point; if  $p$  is not dominated, then it is a part of the skyline. *Divide-and-Conquer (D&C)* [6] approach divides the dataset into several partitions so that each partition fits in memory. Then, the partial skyline in every partition is computed using a main-memory algorithm, and the final skyline is obtained by merging the partial ones. *Bitmap* [18] algorithm encodes in bitmaps all the information required to decide whether a point is in the skyline. *Index* approach [18] organizes a set of lists which contain the minimal coordinates and uses these sorted lists to get the skyline efficiently. *Nearest Neighbor (NN)* [8] performs a nearest neighbor query on the R-tree [7] to find the point with the minimum distance from the beginning of the axis (point  $o$ ). *Branch and Bound Skyline (BBS)* [13] is also based on nearest neighbor search. Specifically, it starts from the root peer of the R-tree, inserts all its entries in a sorted heap, and expands the entries according to their *mindists*, which are the sum of the coordinates. [14] and [21] are two recent papers which concern on how to make use of skylines in subspaces to computer the overall skyline.

These above-mentioned skyline algorithms are all designed for a centralized system. The authors in [3] argue that the skyline operations are most desirable in a distributed web information system and propose a *web based distributed skyline algorithm*. However, this algorithm is for distributed online web services with a centralized control, in spite of the distributed data. In this paper, we consider P2P networks where different data objects can be stored in different peers without any centralized control. Therefore, the *web based distributed skyline algorithm* is not applicable to P2P systems.

### III. EXACT SKYLINE ALGORITHM

#### A. System Model

In SSW [9], semantic space is divided into clusters, each of which takes up a portion of the entire semantic space. Peers are assigned to clusters based on their semantic labels, which are centroids of their largest data clusters. From the above definition we can find that the data in each peer do not strictly fall into the range of the assigned cluster. For those data out of the cluster's range, *foreign indices* are created in the remote peer whose semantic range covers the data value, indicating the location of data objects. With the assistance of foreign indices, peers can send requests to remote peers, asking for a specific data which is actually in the semantic range of that peer. Such guided data migration can guarantee that any cluster in SSW knows whether the data objects belong to its own cluster or not. For a given peer  $p_i$  and the cluster  $C_j$  it belongs to, we call the local data which fall in the range of  $C_j$  and foreign data which are listed in  $p_i$ 's foreign index to be  $p_i$ 's *semantically-local* data. Every peer maintains an auxiliary data structure, which records the semantic range of its semantically-local data. From the above description, it can be easily concluded that a peer's semantic range must be covered by the semantic range of the cluster which it belongs to.

SSW's naming scheme preserves the semantic proximity among clusters as much as possible. By judging from the cluster ID, we can infer the semantic information about the clusters.  $SSW - 1D$  is built upon the naming scheme by linking the clusters in the order of their cluster IDs (short contacts). As well, long contacts, which point to peers randomly chosen in the space, are maintained in  $SSW - 1D$  as well. Figure 2 and 3 illustrate the above definitions. In Figure 2, the whole two dimensional space is partitioned into 11 clusters. Each cluster is given a cluster ID according to the order they are partitioned. Then in Figure 3, these 11 clusters are linked together in the order of their cluster ID.

In addition, each peer maintains a contact list, in which at least one peer in each neighboring cluster is recorded. For example, in Figure 2, the contact list in a peer of  $C_0$  contains entries about peers of  $C_4$  and  $C_2$  since they are adjacent to  $C_0$  in  $y$ - and  $x$ -axis, respectively.

#### B. Problem Definition

Formally, the problem can be described as follows: A skyline query  $Q$  is issued on a peer  $p_0$ , which we denote *initial peer*, in a  $j$ -dimension SSW. Each data object in the network can be represented as a  $j$ -attribute set  $\{v_0, v_1, \dots, v_{j-1}\}$ .  $Q$  is specified to a subset of the  $j$  dimensions. To be more specific,  $Q$  can be expressed as  $Q = \{a_{i_0} : p_{i_0}, a_{i_1} : p_{i_1}, \dots, a_{i_{d-1}} : p_{i_{d-1}}\}$ , where  $i_0, i_1, \dots, i_{d-1} \in \{0, 1, \dots, j-1\}$  and  $d < j$ ;  $p_{i_0}, p_{i_1}, \dots, p_{i_{d-1}} \in \{\min, \max\}$ , indicating the preference over each involved attribute.  $d$  stands for the number of attribute dimension specified in the skyline query. For example, for a 4-dimension SSW  $\{a_0, a_1, a_2, a_3\}$ , a skyline

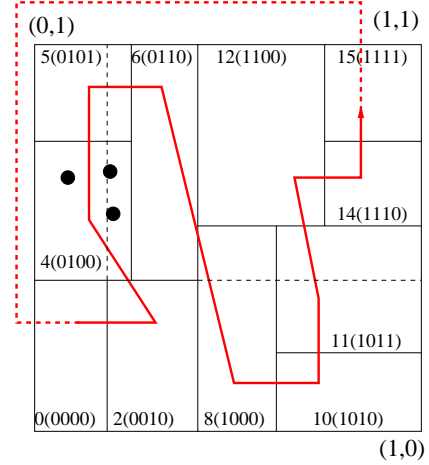


Fig. 2. Cluster ID.

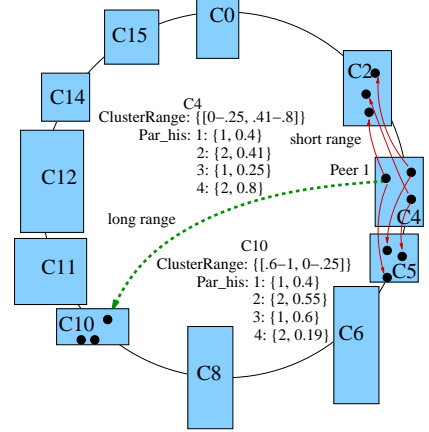


Fig. 3. SSW-1D.

query  $Q = \{a_0 : \min, a_2 : \max\}$  indicates  $Q$  is only related to attribute dimension  $a_0$  and  $a_2$  only. A smaller value in  $a_0$  is preferred and a larger value in  $a_2$  is preferred.

Without loss of generality, we assume a preference for smaller values in every query attribute and all  $j$  attributes are queried. Thus, the definition of  $Q$  can be simplified accordingly:  $Q = \{a_0, a_1, \dots, a_{j-1}\}$ . Now our goal is to find an efficient algorithm to answer such a skyline query in a  $j$ -dimension SSW, issued by a user in peer  $p_0$ . In addition, each SSW dimension semantically ranges from 0 to 1. This assumption does not impair the generality of the problem because any  $j$ -dimension SSW can be mapped into a  $j$ -dimension  $[0, 1]$  SSW without changing the domination relationship between data objects.

#### C. Skyline Algorithm

Similar to the search algorithms used in P2P systems, the P2P based skyline algorithm is executed based on the index structure which is built on top of the peers. An effective index in the P2P system is essential for the skyline algorithm. In this section we propose the exact skyline algorithm, which has the

following precedures.

**Locate the Origin Cluster:** We take advantage of the *naming scheme* in *SSW* to find the cluster whose semantic range covers the optimal attribute value. For example, in a 2-dimension space, we want to find the cluster that covers the origin point  $(0, 0)$ . In *SSW*, such a cluster, which we call *origin cluster* ( $C_o$ ), has a cluster ID of 0. In our algorithm, the initial peer takes advantage of its short contacts to find a peer of  $C_o$ . An important feature of peers in  $C_o$  is that their semantically-local data cannot be dominated by semantically-local data of other clusters. Hence, we can route the query to peers of  $C_o$  to calculate a skyline of  $C_o$ , which is based on all semantically-local data objects of  $C_o$ . The skyline is returned to the initial peer  $p_0$  and can be returned to users immediately because it cannot be dominated by other objects in the network. As well, for all data objects in the skyline of  $C_o$ , we find the data that is nearest to the optimal attribute value (data  $a_1$  in Figure 4), whose value is called *boundary value* in the skyline query ( $v_{bound}$ ), for later pruning.

**Inter-Cluster Pruning:** We make use of the contact list of a peer to forward the skyline query selectively and prune the search space. For a given cluster's semantic range, we take the best values in all dimensions as its representative. If this data object is dominated by  $v_{bound}$ , we say the cluster is dominated by  $v_{bound}$ . Starting from any peer of  $C_o$ , the query is forwarded to peers in neighboring clusters as long as the cluster is not *dominated* by  $v_{bound}$ . Accessed clusters are marked as *visited* to avoid revisits. The peer which receives the query from other clusters is selected as the *representative peer*  $p_{rep}$  of the cluster.  $p_{rep}$  selectively forwards the query again to clusters from its contact list. Such inter-cluster forwarding procedure continues until a  $p_{rep}$  cannot find any cluster in its contact list that is both non-dominated by  $v_{bound}$  and unvisited. In Figure 4,  $C_0$  has three neighboring clusters:  $C_1$ ,  $C_2$ ,  $C_3$ , among which  $C_2$  can be pruned because it is dominated by  $v_{bound}$ , which is the value of data  $a_1$ . All semantically-local data of  $C_2$ , such as  $a_7$ , must be dominated by  $a_1$ .

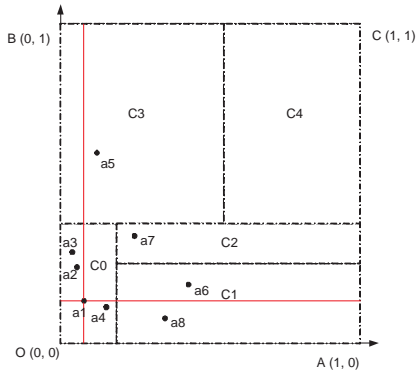


Fig. 4. A 2-Dimension Example for Inter-Cluster Pruning.

**Intra-Cluster Pruning:** Within each cluster that is accessed by the query, we continue to utilize  $v_{bound}$  to prune irrelevant

peers. For every peer in the network, we calculate and record the semantic range of the peer from its semantically-local data objects, as what we do with clusters. Utilizing this range information, if a peer is dominated by  $v_{bound}$ , it can be safely pruned from the query's access path.  $p_{rep}$  of the cluster is responsible for intra-cluster routing and receives skylines of the peers that are accessed in the cluster.

**Skyline Computing:** For each peer that is accessed by the query, it arranges a local skyline from all its semantically-local data objects and returns it to  $p_{rep}$  of the cluster.  $p_{rep}$  collects the skylines and combines them, utilizing an in-memory skyline algorithm. The refined cluster skyline is sent back to  $p_0$ , together with its routing decisions.  $p_0$  runs an in-memory skyline algorithm which keeps accepting incoming skylines and gradually refines it. Also,  $p_0$  keeps track of the routing process. If it finds that all routing branches terminate, it stops listening for incoming skylines and return the results to users.

The formal description of the algorithm is given in Algorithm 1.

#### D. Analysis

In order to prove the correctness of our P2P based skyline algorithm, we first state two lemmas.

**Lemma 1:** *All the pruned clusters and peers should not contain any skyline data.*

**Proof:** The proof is straightforward according to the definition of cluster (peer) domination. Since the boundary value is set to be the nearest data object to the origin of the attribute space, all the other clusters within its dominant region contain the objects with inferior values in the semantic space. In other words, all these objects inside the dominant region are dominated by one known object inside the origin cluster. Thus all these objects are impossible to be skyline objects.

**Lemma 2:** *All final skyline objects cannot be dominated by any objects inside the pruned clusters or peers.*

**Proof:** Without losing generality, we assume a 2-dimension attribute space. Suppose there is an object,  $d_p$ , in a pruned cluster  $C_i$ , dominates one of the returned skyline objects,  $d_s$ . We denote values of  $d_p$  and  $d_s$  as  $(p_x, p_y)$ ,  $(s_x, s_y)$ , respectively.  $v_{bound}$  is represented as  $(v_x, v_y)$ . Suppose we prefer smaller values, then we have  $p_x \leq s_x$  and  $p_y \leq s_y$ . Since  $C_i$  is pruned, we have  $v_x < p_x$  and  $v_y < p_y$ . Hence,  $v_x < s_x$  and  $v_y < s_y$ . In other words, object  $d_s$  is dominated by another object in the network, meaning that it cannot appear in the final skyline. Thus,  $d_p$  cannot dominate  $d_s$ .

Based on the previous two lemmas, our proposed algorithm first finds out the origin cluster and formulates the boundary value, prunes the dominated clusters which do not contain any skyline data, and then searches for skyline objects in the rest clusters. Within a cluster, dominated peers are also pruned. The whole procedure repeats until there is no non-dominated clusters or peers left.

---

**Algorithm 1** Exact P2P skyline algorithm.

---

**Input:**  $p_0$ : Initial Peer $Q$ : Skyline Query**Procedure1(Initial Peer  $p_0$ )**

- 1: formulate the skyline query  $Q$ , with the initial peer information encoded  $Q.id = p_0.id + attributes.id$
- 2: start from  $p_0$ 's short contact, find the origin cluster  $C_o$
- 3: forward  $Q$  to a peer  $p_{rep}$  in  $C_o$
- 4: find a data object which is nearest to the optimal attribute value from all data objects in the skyline of  $C_o$ ; set it to be the boundary value  $v_{bound}$  for  $Q$
- 5: **repeat**
- 6:   receive incoming partial results from peers along the query path
- 7:   refine the current result set  $S$  with newly obtained results
- 8: **until** all query branches terminate

**Procedure2( $p_{rep}$  of an Accessed Cluster)**

- 1: mark the cluster as "visited"
- 2: **for** all neighboring clusters in  $p_{rep}$ 's contact list **do**
- 3:   **if** cluster  $C_i$  is both non-dominated by  $Q$ 's  $v_{bound}$  and not visited **then**
- 4:     route  $Q$  to  $C_i$
- 5:   **end if**
- 6: **end for**
- 7: **for** all member peers in the cluster **do**
- 8:   **if** peer  $p_i$  is not dominated by  $Q$ 's  $v_{bound}$  **then**
- 9:     send  $Q$  to  $p_i$
- 10:   **end if**
- 11: **end for**
- 12: **repeat**
- 13:   receive incoming skylines
- 14:   refine the cluster's skyline with newly obtained results
- 15: **until** all members in the cluster that are chosen to receive  $Q$  have returned their local skylines
- 16: send cluster's skyline to  $p_0$
- 17: send inter-cluster routing decision to  $p_0$

**Procedure3****(Any Peer  $p_i$  That Receives  $Q$  from  $p_{rep}$ )**

- 1: calculate a skyline from its semantically-local data
- 2: return the skyline to  $p_{rep}$  of the cluster

**Output:** the final skyline in  $S$ 

---

## IV. APPROXIMATE ALGORITHMS

In cases where a semantic overlay network does not exist, or where peers in a P2P system cannot acquire or infer necessary semantic information, the previously discussed deterministic algorithm cannot be directly implemented. Whereas some brute forces, such as asking the query to be directed to all clusters with a predefined order, will incur heavy network traffic, especially when the network size becomes extremely large. Instead, We believe what are really desired by users in such scenarios are high-quality results from efficient approximate skyline queries. We explore the possibility of exploiting some effective heuristics to narrow the search scope. Generally speaking, an approximate skyline is built upon a fraction of the entire network. The proposed P2P skyline algorithm is designed to scale well and be robust to cope with the potential node failures. Section IV-A discusses the generic P2P system

model we adopt and some possible brute force algorithms. The approximate P2P Skyline algorithm will be presented in Section IV-B. Alternatives of the approximate algorithm will be discussed in Section IV-C.

## A. System Model

The generic system model makes very few assumptions. Basically, the network is consisted of peers, each of which is connected to some peers in the network. Because each peer has all knowledge and control over the resided data objects, every peer can create a semantic label describing the semantic centroid of the resided data objects. Also, for those peers that one peer knows, the corresponding semantic labels are transferred and stored in the peer's contact list. Different from the model described in Section III-A, although peers in the P2P network can be clustered, the network does not utilize semantic information to create and manage clusters. As a result, any peer (cluster) might contain data objects in the exact skyline.

We follow the same problem definition proposed in Section III. A brute force algorithm can be easily acquired. We take SSW as the example, without utilizing the cluster semantics. Because SSW-1D is constructed on top of a high dimension space, which is connected with short contacts, the initial peer can forward the query to the short contact and wait until the query finishes a loop and returns to the initial peer. Every peer along the cycle collects skyline objects within the cluster (using cluster member links) and forward the query to the next peer in its short contact. It can be easily observed that the query cost, in terms of network traffic, is in proportion with the number of peers in the network.

## B. Single-Path Approximate Algorithm

We call our approximate algorithm the single-path approximate algorithm because each peer tries to find one candidate peer from its contact list to which it can forward the skyline query.

When receiving an incoming skyline query  $Q$ , the initial peer must decide the next candidate peer to which the skyline query is forwarded from its knowledge of contacts. This decision is made upon the semantic labels of peers in the contact list because they are an indication of the attribute values of stored objects. It can be expected that a peer with a semantic label that is closer to the optimal value possibly contain many data objects that are close to an exact skyline. Thus, a chosen candidate peer should have the label value that is best among all contact candidates on the queried attributes. Given the classical hotel example of skyline queries, the current peer will direct the query to a contact peer which has best semantic label in price and beach distance. That is, a contact peer containing a semantic label with lowest price or nearest distance is chosen as the next candidate.

In our algorithm, we choose a score function that takes the semantic distances of all involved attributes into consideration. It can be written as:

$$Score = \min\{(v'_0 - v_0), (v'_1 - v_1), \dots, (v'_{j-1} - v_{j-1})\}$$

In the above definition,  $v'_k$  represents the semantic label value of attribute  $a_k$  of a candidate peer;  $v_k$  represents the semantic label value of attribute  $a_k$  of the current peer. However, it should be pointed out that the above semantic score definition can be varied according to systems' actual requirements. Weight factors can be added to reflect the real influence of these attributes on a decision.

Once the next candidate is decided,  $Q$  is routed to it. At the same time, the current peer is marked as visited to avoid unnecessary revisits by the query. Such a procedure continues, creating a query execution path. Every peer along the path calculates a local skyline from its local data, using a centralized skyline algorithm, and returns the results to the initial peer. Thus, the initial peer can progressively receive the results from the network. A skyline combination subroutine (utilizing centralized skyline algorithms) continues accepting incoming results and produces a gradually refined skyline. This algorithm terminates when it finds out that there is no better candidate contact than the current peer. A stop signal is issued by the current peer to the initial peer to terminate the query processing. The formal description of the algorithm is presented in Algorithm 2.

---

**Algorithm 2** Single-path skyline algorithm.

---

**Input:**  $p_0$ : Initial Peer

$Q$ : Skyline Query

**Procedure1(Initial Peer  $p_0$ )**

- 1: formulate the skyline query  $Q$ , with the initial peer information encoded  $Q.id = p_0.id + attributes.id$
- 2: calculate a skyline from local data and store it as a partial result  $S$  for further refinement
- 3:  $p_0.visited = 1$
- 4: find the best candidate peer ( $p_1$ ) from  $p_0$ 's contact list in the queried attributes (using the score function)
- 5: **if**  $p_1$  is better than  $p_0$  **then**
- 6:     forward  $Q$  to  $p_1$
- 7: **else**
- 8:     end the procedure
- 9: **end if**
- 10: **repeat**
- 11:     receive incoming partial results from the peers along the query path
- 12:     refine the current result set  $S$  with the newly obtained results
- 13: **until** a termination signal from query path is received

**Procedure2**

**(Other Peers  $p_i$  along the Query Path)**

- 1: calculate a skyline from local data of  $p_i$  and return it to  $p_0$
- 2:  $p_i.visited = 1$
- 3: find the best candidate peer ( $p_j$ ) from  $p_i$ 's contact list in the queried attributes (using the score function)
- 4: **if**  $p_j$  is better than  $p_i$  **then**
- 5:     forward  $Q$  to  $p_j$
- 6: **else**
- 7:     a termination signal is sent to  $p_0$
- 8: **end if**

**Output:** the final skyline in  $S$

---

For the ease of understanding we depict how one peer decides the next candidate peer in the algorithm, using the

hotel example. Suppose the current peer has a semantic label of (101, 66) and it maintains a contact list in which the semantic labels are given in Table I. Because a lower price and a shorter distance to a beach are desired to most travelers, the optimal values for price and distance attributes are both 0. The scores for all contact peers are also given in the final column:

| Peer ID | Price(\$) | Distance to Beach(mile) | Score |
|---------|-----------|-------------------------|-------|
| A       | 79        | 65                      | -22   |
| B       | 182       | 65                      | -1    |
| C       | 103       | 70                      | 2     |
| D       | 69        | 84                      | -32   |

TABLE I

AN EXAMPLE OF THE SINGLE-PATH ALGORITHM PROCESSING.

From the above data, our algorithm chooses  $D$  as the next routing peer because it has a lowest score value in the list.

The correctness of this peer-to-peer skyline algorithm depends on the correctness of the local centralized skyline algorithm used in every peer in the network. As well, its convergence can be proven by considering the limited scope of a given network and the identification of visited peers along the query path. That is, a peer cannot be visited more than once during a query's execution period. In addition, because of the distribution of long contacts, the algorithm can quickly find a better non-local peer and direct the query there. Thus, it can be expected that the algorithm will terminate quickly in spite of the huge amount of peers in the network.

### C. Discussions and Improvements

The single-path skyline algorithm can be applied to heterogeneous P2P network systems. However, under some specific circumstances, the effect of the algorithm can be enhanced by using some alternatives.

The heuristics we adopt in the single-path algorithm calculates the semantic distances of the involved attributes and chooses a single peer from a candidate contact list. However, considering the following case in the hotel scenario, there are two hotel agents, A and B, in the candidate list. The semantic label of A is cheaper, while B is nearer to a beach. If we simply choose one peer from the list, say A, and direct the query to it, it is often possible that B will not be considered again in the query path even if it contains many hotel records that are near to a beach. Therefore, an important portion of a good skyline is neglected. Another case comes as choosing a destination from a three-peer list,  $A$ ,  $B$  and  $C$ , where  $A$  is optimal in price,  $B$  is optimal in spatial location, and  $C$  is very good in price and location in their semantic labels. In the previous heuristics function, probably  $C$  will be chosen as a candidate, throwing  $A$  and  $B$  away, which will greatly decrease the quality of the approximate skyline.

Observing the existence of such scenarios, we can use a different heuristics in the algorithm to improve the skyline quality. The alternative score function is shown as follows:

$$Score = \{(v'_0 - v_0), (v'_1 - v_1), \dots, (v'_{j-1} - v_{j-1})\}$$

In the above function, all the parameters are the same as what are defined in Section IV-B. Hence, the score function returns a  $j$ -tuple set instead of a single result. Accordingly, up to  $j$  candidate destination peers,  $n_0, n_1, \dots, n_{j-1}$ , can be chosen (for which we call the alternative approach a multi-path skyline algorithm), where  $n_k (k = 0, 1, \dots, j - 1)$  represents the peer in the contact list with the best value in attribute  $a_k$ . If none of the candidate peers is better than the current peer in attribute  $a_k$ ,  $n_k$  is set to be *null*. The initial peer is informed with the routing decision so that it can keep track of the query path branches. Using this approach, more peers will be visited during the execution of a skyline query, bringing up the accuracy of returned skyline.

Suppose we have the hotel records as listed in Table II, using multi-path algorithm will generate the score table as follows:

| Peer ID | Price(\$) | Distance to Beach(mile) | Score   |
|---------|-----------|-------------------------|---------|
| A       | 79        | 65                      | -22, -1 |
| B       | 73        | 73                      | -28, 7  |
| C       | 88        | 59                      | -13, -7 |
| D       | 182       | 65                      | 81, -1  |
| E       | 103       | 70                      | 2, 4    |
| F       | 69        | 84                      | -32, 18 |
| G       | 90        | 68                      | -11, 2  |

TABLE II  
AN EXAMPLE OF THE MULTI-PATH ALGORITHM PROCESSING.

In Table II, the algorithm will select two candidate peer  $F, C$  from the contact list because one is optimal in price and the other is optimal in distance. So the current peer will forward the query to  $F$  and  $C$ , creating two query sub-paths.

Termination condition of the single-path algorithm can be modified as well. Instead of passively waiting and receiving data with acknowledge messages, the initial peer can actively observe and decide the termination timing. For example, as the initial peer finds a certain number of peers have been traversed by the algorithm, or a certain amount of skyline data objects have been collected, it can send a stop message to the current active peers in the query path. Correspondingly, the peers in the query path should work in a synchronous method, waiting until an acknowledgement reaches before forwarding the query to the next candidate.

## V. EVALUATION

Extensive simulations are performed to test the effectiveness and efficiency of our proposed skyline algorithms. We will briefly describe the design and implementation of the evaluation in Section V-A. The metrics adopted in comparisons are presented in Section V-B. Finally, categorized evaluation results are presented and discussed in Section V-C.

### A. Simulation Setup

We will run the algorithms in different settings of SSW, which are listed in Table III, together with default values. We choose the value range for each attribute to be within  $[0, 1]$ , with a smaller value preferred by users. If not specified in the following subsections, the tests are performed under the standard setting.

| SSW Parameter                    | Range     | Default Value |
|----------------------------------|-----------|---------------|
| Number of peers in a network     | 128 - 16K | 1024          |
| Size of peer clusters            | 8 - 40    | 16            |
| Number of data records in a peer | 10 - 500  | 100           |
| Attribute dimension              | 2 - 5     | 2             |
| Skewness of data dataset - Zipf  | 0.0 - 1.0 | 0             |

TABLE III  
SIMULATION SETTINGS.

The meanings of most of the parameters in the above table are self-explanatory. To describe the data semantic distribution among peers, SSW associates a Zipf-distribution to the skewness of data, where 1 represents a high skewness while 0 represents a uniform distribution. To avoid the possible uneven aggregation of query peers, our experiment workload initiates skyline queries from all the peers in the network and calculates the mean performance.

### B. Metrics

What are most critical to an online skyline algorithm (exact or approximate) working in a P2P network are the query execution time, and the network traffic caused by a query, which can be described with the following metrics:

**Search Path Length (Hops)** is the average number of network hops visited during the query execution.

**Search Cost** is the network traffic caused by the execution of the algorithms, which includes partial results delivery and auxiliary messages among peers.

To an exact skyline algorithm, we need to observe the proportion of accessed regions, compared with the entire P2P network size. We propose two metrics for this purpose:

**Cluster Access Percentage** is the percentage of accessed clusters compared with the number of clusters in the network.

**Peer Access Percentage** is the percentage of accessed peers compared with the number of peers in the network.

For approximate skyline algorithms, we are interested in studying how close an approximate skyline is to an exact one. We propose the following metric to serve this aim:

**Result Quality** returns the area between an approximate skyline with a completely exact one that takes all the data objects in the network into consideration. Figure 5 illustrates the concept in a 2-dimension space, where the dotted curve represents the approximate skyline, the solid curve represents the exact skyline, and the dark area between the two curves indicates the approximation degree.

In the implementation, we use a random number generator to generate a large amount (10000) of uniformly distributed points in the attribute space. After that, we calculate the

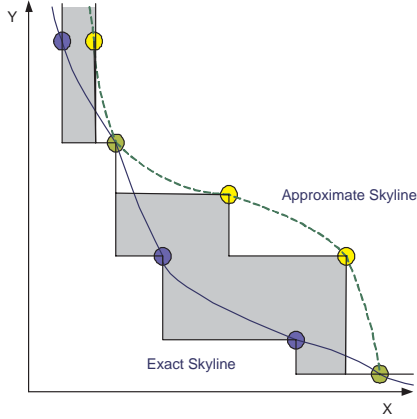


Fig. 5. The Result Quality of the Skyline.

percentage of the generated points which fall between the two skylines to approximate the area percentage over the entire valid attribute space. A smaller result value indicates the region between two skylines is small. That is, a better degree of approximation is obtained.

### C. Results

1) *Scalabilities*: In terms of scalabilities, we study the effects of network size, number of data objects per peer and data attribute dimensionality to our proposed algorithms.

First we vary the number of peers in SSW from  $2^7$  to  $2^{14}$ , with other parameters remained in their default values.

From Figure 6(a) and 6(b), we can see the multi-path algorithm has a higher response time and a larger traffic cost than the other two algorithms, which are not very sensitive with the network size, remaining in a very limited scope. Because the dimension of data remains static for all data objects in the network, the hop diagram follows a similar curve with the traffic messages. The multi-path algorithm asks for more hops and messages than other approaches because in a dimension setting of two, a more branch will possibly originate from any peer along the query path. In terms of returned skyline quality, Figure 6(c) shows that multi-path algorithm outperforms single-path algorithm, basically staying in the range between 0 and 5 out of 10000 uniformly distributed two-dimension points, whereas the single-path algorithm stays in the range from 40 to 450. Considering the vast network size and the limited search hops, the approximation level is excellent. Figure 6(d) suggests the access percentage of the exact algorithm decreases as the peer number grows up. This is because as the size of the network increases, more clusters are created. And the proportion of clusters that are small in any dimension decreases. Therefore, more clusters can be easily pruned by the algorithm, showing the exact algorithm can perform well for huge P2P networks.

Next, we set the network size to be a constant value of 1024 and study how the number of data objects maintained at each peer (from 10 to 500) can influence the performances. As the

number of data objects increases, there is an increase in the number of local skyline data maintained in each peer accordingly. Thus, more partial results need to be returned to the initial peer, which contributes to the increase in the message, depicted in Figure 7(b). However, because the network size and data distribution remains unchanged, the average query lengths are not affected too much, which is shown in Figure 7(a). Similarly from the first experiment, multi-path approach outperforms single-path in Figure 7(c), with the expense of more hops and network traffics. The quality of returned skyline even improves as more data is stored in peers because the data density gets higher, making the semantic distance between data objects lower. Figure 7(d) shows that with more data stored, the cluster access rate remains the same, while there is an increase in peer access rate. This is because as the number of data objects in a single peer is large, more overlapping can be found for peers in a cluster. Therefore, it is more difficult to find an irrelevant peer within a cluster that can be safely pruned.

Other than network size and number of data objects, data dimension has much influence on the performance of the algorithms. From the description of the algorithms in previous sections, we can see the number of dimensions can greatly affect the number of accessed clusters (peers) for the exact algorithm and decide the number of query branches of the multi-path approximate algorithm. In our experiments, we change the number of data dimension from 2 to 5.

The evaluation results suggest that in terms of hops and network traffic, our proposed algorithms grows gradually up with the dimension. In Figure 8(a), we can find that when the dimension increases to five, the exact algorithm can terminate within 600 hops, nearly  $\frac{1}{2}$  of the entire network; while the single-path approximate algorithm can terminate within 400 hops, nearly  $\frac{1}{3}$  of the entire network. The increase in network traffic is more obvious because more data attributes need to be transferred. Also, with the number of visited peers increases with the dimension, the approximate quality of the single-path approach improves dramatically (almost an ideal one when the dimension reaches 5), which is shown in Figure 8(c). Figure 8(d) suggests that more clusters (peers) are involved with high dimensions, because a larger proportion of clusters have semantic labels that are close to optimal values in at least one dimension.

2) *Data Distribution*: A study on data distribution is critical to P2P query algorithms because actual data distribution is not likely to be uniform. It is frequent to find that relevant data objects are grouped together, stored in a peer or some neighboring peers. Therefore, if it can be proved that a portion of peers in the network have some good data objects for skylines, they become the *sweet region* in the network that contributes a lot to a good skyline. Now we examine and compare the performance of these algorithms with different data distributions.

From Figure 9(a) and 9(b), we can see all the algorithms

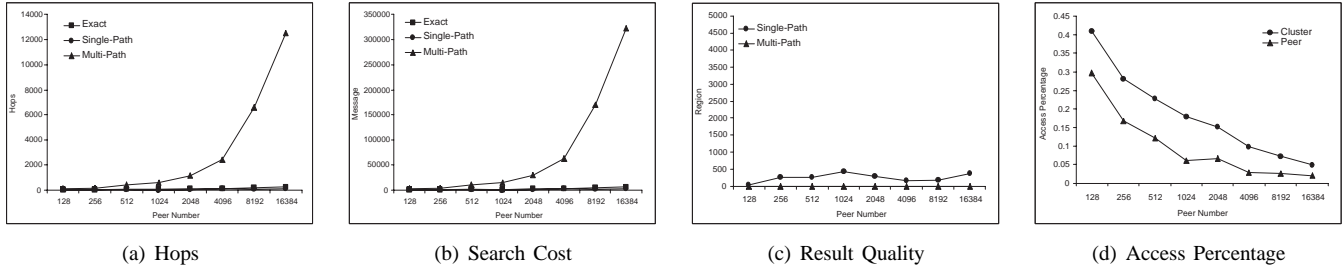


Fig. 6. Comparison Based on Different Number of Peers in the Network.

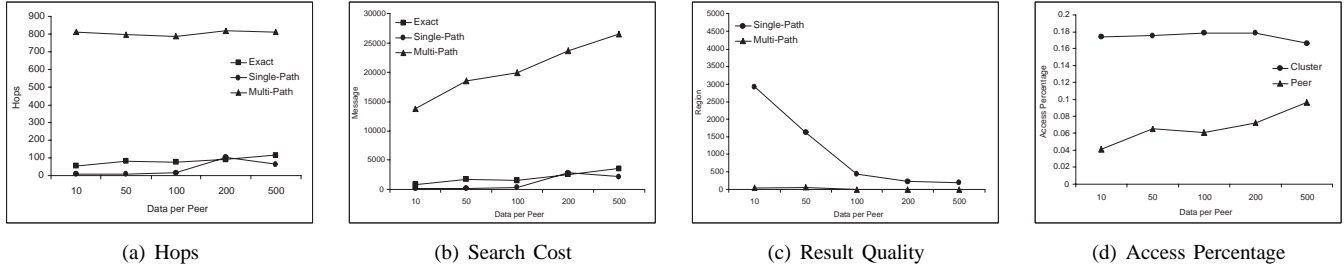


Fig. 7. Comparison Based on Different Number of Data Records in the Peer.

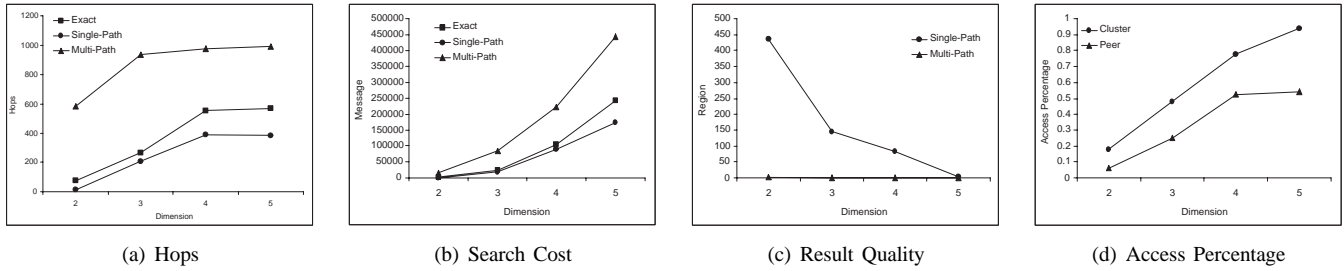


Fig. 8. Comparison Based on Different Data Dimensions.

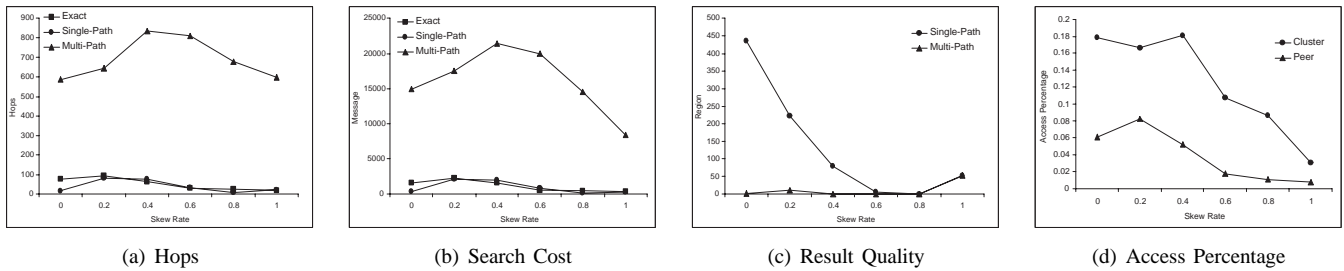


Fig. 9. Comparison Based on Different Skewness.

can terminate quickly in spite of the data distribution. As well, the response time decreases when the data is highly skewed, because the highly clustered data region can be either easily pruned or be easily detected to prune other sparse regions. As a result, we can see from Figure 9(c) that the single-path approach is very applicable to highly skewed data distribution. The approximate skyline quality generated by the single-path algorithm is almost the same as the multi-path approach, with a smaller cost. Figure 9(d) tells us the same effect happens to the exact algorithm: highly skewed data distribution makes

the job of eliminating unnecessary clusters and peers easier, reducing the corresponding access percentage.

3) *Clustering Effects*: The exact algorithm runs on the basis of SSW semantic clusters. We can study the effects of different cluster sizes to the algorithm's performance. From the description of the algorithm, we can infer that if the number of peers in the network remains the same, smaller clusters have small-scale semantic regions. Pruning is more likely to happen among clusters. On the other hand, although it is difficult to entirely eliminate a large-scale cluster from the candidate

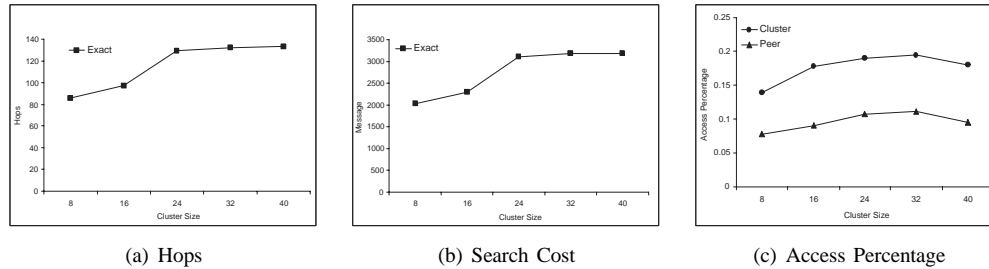


Fig. 10. Comparison Based on Different Cluster Sizes.

cluster list, our algorithm can still prune irrelevant peers within a cluster. The above expectation can be proven by Figure 10. It can be observed from Figure 10(a) and 10(b) that when the cluster is small, the hops and messages are relatively small as well. This is because the "early" pruning on the cluster basis avoids some traffic between peers within a cluster.

## VI. CONCLUDING REMARKS

Existing skyline algorithms are mostly designed for centralized databases or for distributed data sources with a centralized control. In this paper, we consider a fundamentally different environment that has not yet been explored: P2P systems where all peers share an identical role in functions. Based on the structure of semantic overlay networks, our proposed exact skyline algorithm detects and prunes clusters based on their semantic boundaries. In addition to the exact skyline algorithm which runs on a semantic-clustered overlay network, we propose a routing-based approximate skyline algorithms that utilize effective heuristics to control the search scope.

As for the future work, we are going to design a skyline-specific distributed index framework on top of the peers in semantic overlay networks to facilitate querying processing. Many technical problems such as index distribution and maintenance will be examined under this framework.

## VII. ACKNOWLEDGEMENTS

This research work is partially supported by National Science Foundation Grant IIS-0534343.

## REFERENCES

- [1] Gnutella. Website at <http://gnutella.wego.com/>.
- [2] Napster. Website at <http://www.napster.com/>.
- [3] W. Balke, U. Guntzer, and J. X. Zheng. Efficient distributed skylining for web information systems. In *Proc. of the 9th International Conference on Extending Database Technology (EDBT 2004)*, Heraklion, Crete, Greece, 2004.
- [4] W. Balke, W. Nejdl, W. Siberski, and U. Thaden. Progressive Distributed Top k Retrieval in Peer-to-Peer Networks. In *Proceedings of the 21st International Conference on Data Engineering (ICDE'05)*, Tokyo, Japan, April 2005.
- [5] Ashwin R. Bhambe, Mukesh Agrawal, and Srinivasan Seshan. Mercury: supporting scalable multi-attribute range queries. In *SIGCOMM '04*, 2004.
- [6] S. Borzsonyi, D. Kossmann, and K. Stocker. The skyline operator. In *IEEE Conf. on Data Engineering*, pages 421–430, Heidelberg, Germany, 2001.
- [7] Antonin Guttman. R-trees: a dynamic index structure for spatial searching. In *SIGMOD '84: Proceedings of the 1984 ACM SIGMOD international conference on Management of data*, pages 47–57, 1984.
- [8] D. Kossmann, F. Ramsak, and S. Rost. Shooting stars in the sky: An online algorithm for skyline queries, 2002.
- [9] M. Li, W. Lee, and A. Sivasubra. Semantic small world: An overlay network for peer-to-peer search. In *Proc. of 12th IEEE International Conference on Network Protocols*, Berlin, Germany, October 2004.
- [10] Mei Li, Wang-Chien Lee, and A. Sivasubramaniam. Neighborhood signatures for searching p2p networks. In *Proceedings of Seventh International Database Engineering and Applications Symposium (IDEAS'03)*, pages 149–158, 2003.
- [11] Bin Liu, Wang-Chien Lee, and Dik Lun Lee. Supporting complex multi-dimensional queries in p2p systems. In *ICDCS '05*, 2005.
- [12] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and replication in unstructured peer-to-peer networks. In *Proc. of ACM International Conference on Supercomputing*, pages 84–95, June 2002.
- [13] Dimitris Papadias, Yufei Tao, Greg Fu, and Bernhard Seeger. An optimal and progressive algorithm for skyline queries. In *SIGMOD 2003: Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 467–478, 2003.
- [14] Jian Pei, Wen Jin, Martin Ester, and Yufei Tao. Catching the best views of skyline: A semantic approach based on decisive subspaces. In *VLDB '05*, 2005.
- [15] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A scalable content addressable network. Technical Report TR-00-010, Berkeley, CA, 2000.
- [16] Antony Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, 2001.
- [17] Ion Stoica, Robert Morris, David Karger, Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable Peer-To-Peer lookup service for internet applications. In *Proceedings of the 2001 ACM SIGCOMM Conference*, pages 149–160, 2001.
- [18] Kian-Lee Tan, Pin-Kwang Eng, and Beng Chin Ooi. Efficient progressive skyline computation. In *VLDB '01: Proceedings of the 27th International Conference on Very Large Data Bases*, pages 301–310. Morgan Kaufmann Publishers Inc., 2001.
- [19] C. Tang, Z. Xu, and S. Dwarkadas. Peer-to-peer information retrieval using self-organizing semantic overlay networks, 2002.
- [20] Beverly Yang and Hector Garcia-Molina. Improving search in peer-to-peer networks. In *ICDCS '02: Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS'02)*, page 5, 2002.
- [21] Yidong Yuan, Xuemin Lin, Qing Liu, Wei Wang, Jeffrey Xu Yu, and Qing Zhang. Efficient computation of the skyline cube. In *VLDB '05*, 2005.
- [22] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, UC Berkeley, April 2001.