Wang B, Yang XC, Wang GR *et al.* Continually answering constraint *k-NN* queries in unstructured P2P systems. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 23(4): 538–556 July 2008

Continually Answering Constraint *k-NN* Queries in Unstructured P2P Systems

Bin Wang[1] (王 斌), Xiao-Chun Yang[1] (杨晓春), Guo-Ren Wang[1] (王国仁), Ge Yu[1] (于 戈), Lei Chen[2] (陈 雷) X. Sean Wang[3] (王晓阳), and Xue-Min Lin[4] (林学民)

[1] *College of Information Science and Engineering, Northeastern University, Shenyang 110004, China*

[2] *Department of Computer Science, The Hong Kong University of Science and Technology, Hong Kong S.A.R., China*

[3] *Department of Computer Science, University of Vermont, Vermont, U.S.A.*

[4] *Department of Computer Science, The University of New South Wales, Australia*

E-mail: {binwang, yangxc, wanggr, yuge}@mail.neu.edu.cn; leichen@cs.ust.hk; xywang@cs.uvm.edu; lxue@cse.unsw.edu.au

Received May 7, 2007; revised February 9, 2008.

Abstract We consider the problem of efficiently computing distributed geographical *k-NN* queries in an unstructured peer-to-peer (P2P) system, in which each peer is managed by an individual organization and can only communicate with its logical neighboring peers. Such queries are based on local filter query statistics, and require as less communication cost as possible, which makes it more difficult than the existing distributed *k-NN* queries. Especially, we hope to reduce candidate peers and degrade communication cost. In this paper, we propose an efficient pruning technique to minimize the number of candidate peers to be processed to answer the *k-NN* queries. Our approach is especially suitable for continuous *k-NN* queries when updating peers, including changing ranges of peers, dynamically leaving or joining peers, and updating data in a peer. In addition, simulation results show that the proposed approach outperforms the existing Minimum Bounding Rectangle (MBR)-based query approaches, especially for continuous queries.

Keywords unstructured P2P, *k-NN* queries, answering queries, constraints

1 Introduction

Due to their importance in many applications in a variety of domains, *k-NN* queries have been extensively studied[1−5]. An often used mechanism that provides much needed retrieval efficiency is a centralized index. However, for *k-NN* queries in a distributed environment[6−9], especially in an unstructured P2P environment, centralized indexing is not a practical solution. The following example shows why new techniques are needed and hence motivates the work of this paper.

Example 1. Consider a tsunami alarm system for a certain area, e.g., the bay area of Indonesia. Detection of a tsunami in many cases needs data from areas that go across multiple nations. Assume the nations establish a logical cooperative network, in which for any two nations, they have either direct cooperative relationship, represented as logical neighbors in the network, or indirect cooperative relationship if their logical neighbors can cooperate. Each nation autonomously maintains a set of observation stations to monitor her own sea area and can request data from her cooperative nations. Notice that it is not necessary that two logical neighbors are geographically bordered. For this application, a user often needs to pay special attention to a set of k-closest observation stations in a particular location such that their sensed values satisfy a specified condition, for instance, the temperatures are greater than 80 degrees. Observation stations in the interested area may belong to different nations. Therefore, those autonomous nations need to cooperate in answering a continuous distributed *k-NN* query efficiently. In this application, global indexing is not available for the *k-NN* query, either, since data values change continuously.

Example 1 illustrates a new constrained continuous *k-NN* search problem over an unstructured P2P environment, which can be described as follows.

Given an unstructured P2P system, in which each autonomous peer chooses logical neighbors that are not necessarily its physical neighbors. Assume that each peer maintains a "dynamic" table that contains a set of locations and the associated attribute values, with the

Regular Paper

Supported by the Program for New Century Excellent Talents in Universities (Grant No. NCET-06-0290), the National Natural Science Foundation of China (Grant Nos. 60503036, and 60773221), the National High-Tech Development 863 Program of China (Grant No. 2006AA09Z139), and the Fok Ying Tong Education Foundation Award (Grant No. 104027).

condition that the locations must be within the spatial region covered by the peer. Here, dynamic means the location and value information are frequently updated. Given a query location and a value predicate, a constrained continuous *k-NN* search over such a P2P setting is to find the k nearest locations whose associated values satisfy the value predicate with the minimum overall communication cost.

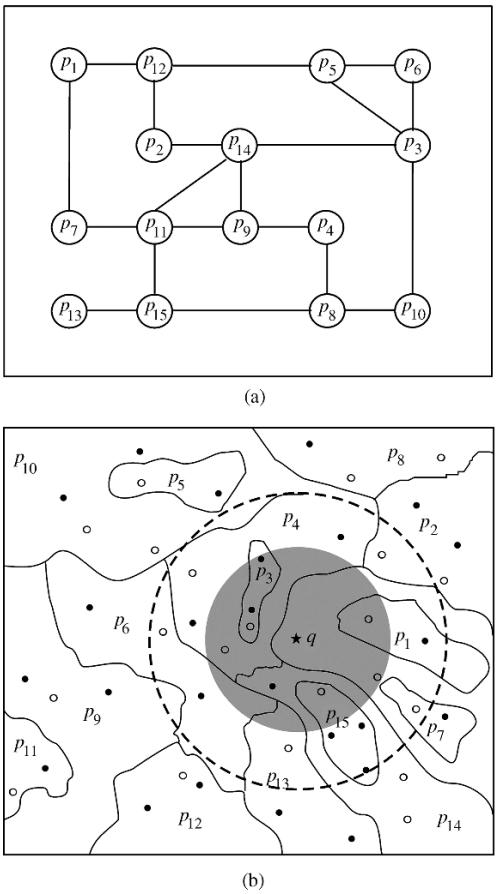


Fig.1. Example of continuous top-k queries in autonomous cooperative peers. (Each black node in (b) represents a data item in the peer satisfying query predicates.) (a) Logical structure of peers. (b) Geometrical peer regions.

We demonstrate the above search problem in Fig.1. As shown in Fig.1(a), 15 autonomous peers are connected as a logical communication graph and each autonomous peer can only communicate with its logical neighbors. Fig.1(b) shows the spatial regions covered by all the peers, and the observation stations appearing in them. Assume that now a peer, p_1, submits a query q to find 3 closest observation stations whose observed temperatures are greater than 80 degrees. The three black dots in the shaded area shown in Fig.1(b) are the answers to q. To find these observation stations, a flooding approach may be used. In this approach, p_1 sends q to all its (direct and indirect) cooperative peers, and each peer returns its own 3-*NN* data (if any), which are then merged to produce the overall 3-*NN* results. Obviously, this approach may waste a lot of bandwidth since many data sent are unnecessary, thus is not a desirable solution in such a distributed environment.

In summary, a continuous constrained *k-NN* search with value predicates in an unstructured P2P system has the following features. These features pose unique challenges in comparison with the distributed *k-NN* problems on P2P that have been considered by other authors[10,11].

1) *Topology mismatch between the P2P logical overlay network and physical underlying network.* For example, if a peer p_1 wants to probe a physically close-by peer p_3, it has to transmit a *k-NN* query q through a path from p_1 to p_3. Fig.1(a) shows that there are several paths from p_1 to p_3. Comparing the two paths, $p_1 \rightarrow p_{12} \rightarrow p_5 \rightarrow p_3$ and $p_1 \rightarrow p_{12} \rightarrow p_2 \rightarrow p_{14} \rightarrow p_3$, we find the former is the shortest path which incurs least communication cost $3(\alpha + \beta \cdot k)C_0$, where 3 is the number of edges in the path, α is the data size of the query q, β is the data size of each returned data, and C_0 is the communication cost between every two logical neighbor peers.

However, Fig.1(b) shows that p_3 only provides 2 out of 3 nearest neighbors, and p_1 must probe more peers to get the remaining one. Probing p_{14} to get that result will incur another $3(\alpha + \beta \cdot k)C_0$ communication cost. If p_1 probes the second path directly, then it only incurs $4(\alpha + \beta \cdot k)C_0$ communication cost to get all the answers to q. Ideally, to get the query results, we hope to probe as few peers as possible while it incurs as little communication cost as possible.

2) *Data in each peer is maintained autonomously in an unstructured P2P system.* Unlike structured P2P systems, in an unstructured P2P system, we have no global information about how a peer organizes all its data.

3) *Value predicate may contain conditions on values that are independent of the location.* A peer with shorter distance to q may not contain data that can answer q. Re-examine the query q: "Find 3 closest observation stations whose sensed temperatures are greater than 80 degrees." If a peer with shorter distance to q cannot answer q, it does not mean that another peer with longer distance to q cannot answer q.

4) *The unstructured P2P system changes dynamically.* Updates in the system are listed as follows, which requires technique for continuously processing a *k-NN* query.

- Peers can join or leave the system at any time.
- The region covered by a peer can expand and shrink

dynamically.

• Data maintained in each peer are changed autonomously.

The existing *k-NN* query approaches can be categorized into *radius-convergence strategy*, in which the probe radius is shrunk gradually until all peers in the probe circle have been probed. In this paper, we propose a new framework to address the above problem. Our frame work is based on a dominate relationship model (DM in short), which will be used to filter out unqualified peers during the query processing. On the basis of the DM, we introduce two directions to efficiently probe as few as possible peers to answer a new constrained *k-NN* query or a constrained *k-NN* query from beginning (restart the continuous query due to the environment change) with minimized communication cost. In order to adapt to continuity property of continuous query, we propose a novel approach that can incrementally maintain the query results based on the data histograms of the peers. Our contributions of this paper are summarized as follows.

• We propose a new framework for processing continuous *k-NN* queries in an unstructured P2P system.

• We give a novel filtering mechanism to reduce the communication cost and effectively terminate the search. Instead of using existing radius-convergence strategy, we propose a *radius-expansion strategy* to efficiently get *k-NN* results. We also give detailed complexity analysis of our algorithms.

• We analyze system updates in an unstructured P2P system, and propose techniques for continuously processing *k-NN* queries.

The rest of the paper is organized as follows. Our framework for continuous *k-NN* queries are presented in Section 2. In Section 3, a novel filter model, called domination model, is proposed to efficiently prune peers. Section 4 proposes radius-expansion strategy for searching *k-NN* queries with least communication cost. In Section 5, we propose histogram for optimal probing peers. In Section 6, we present our novel techniques to incrementally maintain continuous query results. In Section 7, we discuss the related work. Experimental results and performance studies are discussed in Section 8. Finally, Section 9 concludes the paper.

2 Problem Definition

In this section, we formally define the query problem studied in this paper. We assume a set of logically connected cooperative peers, each of which covers a spatial region. We use a non-directed graph $G = (P, E)$ to model the logical connections, where P is a set of vertices representing the peers and E a set of edges expressing the logical connections between the peers.

For a peer $p \in P$, we use $D(p)$ to denote data set maintained by p. Each item in $D(p)$ maintained by p has two kinds of attributes, namely location attributes and non-location attributes. Hence, we denote each data item d in $D(p)$ as a pair $\langle \boldsymbol{v}_l(d), \boldsymbol{v}_n(d) \rangle$, where $\boldsymbol{v}_l(d)$ ($\boldsymbol{v}_n(d)$, resp.) is a value vector of the location (non-location resp.) attributes of d. We can construct a convex hull using values of the location attributes in $D(p)$. The convex hull is called spatial region of p, denoted by $R(p)$. That is, for each d in $D(p)$, $\boldsymbol{v}_l(d)$ must be contained in the region $R(p)$.

A *k-NN* query q is composed of two parts, location value and non-location value predicates, denoted as $\langle \boldsymbol{v}_l(q), q_c \rangle$, respectively. A non-location value predicate q_c returns, when applied to a non-location value vector $\boldsymbol{v}_n(d)$, true or false. We define the distance between a query q and data d as the following.

Definition 2.1 (Distance Between Query and Data). *Given a query q and data d, let the distance between q and d be*

$$dist(q,d) = \begin{cases} dist(\boldsymbol{v}_l(q), \boldsymbol{v}_l(d)), & \text{if } \boldsymbol{v}_n(d) \text{ is true,} \\ \infty, & \text{otherwise,} \end{cases}$$

where dist can be any of the L_p-norm.

Given a query q and a peer p, we use $minDist(q,p)$ ($maxDist(q,p)$, resp.) to express the minimum (maximum, resp.) distance between q and all the data items in $R(p)$.

We assume that each peer has pre-knowledge of spatial regions covered by all the other peers. This assumption is reasonable since there are a variety of techniques to get such statistical information in dynamic network environment. For example, distance-vector-routing[12] is one of mature techniques. Each node in the network can exchange information with its neighbors and can get the statistical information from the whole network in a few steps. A peer can also collect data from different peers and derive this knowledge in an incremental manner. In fact, it is not necessary to get all initial regions from all peers. The initial regions of peers (could be partial peers) can help to do query estimation. That is, our technique is used to estimate a "good" (but might not be precise) query radius to get the closest k results over unstructured P2P network, so that we could save more communication cost. In the initial regions, we can get some clue to estimate the query radius. Then, we submit the *k-NN* query along the logical structure of the P2P network to collect results. We also propose a histogram-based approach to incrementally collecting statistical information in Sections 5 and 6.

In this paper, we can adopt such techniques to collect

necessary statistical information, since our emphasis focuses on how to use those pieces of statistical information, like minimal and maximal distances between the query q and the other peers, to continually process *k-NN* queries.

We adopt the well-known *local filtering mechanism*[13] to calculate *k-NN* answers. Local filtering mechanism pushes a query q down into the network from the query peer p_q in a query dispatch phase, and the resulting values are routed up along the reversed query paths and eventually to p_q in a collection phase. For each peer p that receives q, it computes the k nearest data items of q from the union of $D(p)$ and the data items returned from its neighbor peers. Then p returns these k nearest data items to the peer that sent the query q to it. We assume that each query q and each data item has a similar size. If n peers are used for processing q, then the total amount of communication cost in sending q and receiving data items is $n(\alpha + \beta \cdot k)C_0$, where α is the data size of the query q, β is the data size of each returned data, and C_0 is the communication cost between every two logical neighbor peers.

Now, we describe the continuous *k-NN* query on a set of connected (autonomous) peers. *Formal Problem Statement*: assume we have a set of logically connected peers $p_1, \ldots, p_n$, each of which manages a set of data items, denoted by $D(p_i)$, with locations of each data item in $D(p_i)$ being in the spatial region $R(p_i)$. Given a continuous query q issued by a peer p_i, continually search k data items $\{d_1, \ldots, d_k\}$ among the data items in $D(p_1) \cup \cdots \cup D(p_n)$ such that there does not exist any data item d satisfying the condition $dist(q, d) \leqslant dist(q, d_h)$ for some $1 \leqslant h \leqslant k$. We aim at minimizing the total communication costs.

Table 1 summarizes the notations throughout the paper.

Table 1. Notations in the Paper

Notations	Description
q	*k-NN* query
$R(p)$	Region of a peer p
$dist(q, d)$	Distance between query q and data d
$minDist(q, p)$	Minimal distance between query q and peer p
$maxDist(q, p)$	Maximal distance between query q and peer p
$G(P, E)$	Logical structure of a set of peers P, where E is the set of neighborhood
r_k	Distance between q and the k-th closest data item
r_p	Query radius of probe circle
r_a	Query radius of estimated result circle

3 Domination Model

The simplest approach to answering the continuous constrained *k-NN* search is using flooding. However, it has the following three shortcomings.

- It is very costly in terms of communication cost.
- It probes all the peers in the system. However, in most of the cases, we only need a few peers to get the *k-NN* data to the query.
- It starts the same query from beginning without reusing the previous query results. However, for a continuous query, it is possible to reuse the previous query results with little change on the updated data.

In this section, we propose a *domination model* (or DM) to guide the probing of peers in order to significantly reduce the communication cost by limiting the participating peers to a minimum number possible. The DM together with the approaches proposed in Section 4 are used to address the first two shortcomings of flooding. The third shortcoming of flooding is tackled in Section 5.

The intuition of the domination model comes from the following simple facts: if we know an upper bound r^u of the distance from the query point q to the k-th data item in the query results, then a peer p' does not need further probing (for possible answers) if $r^u \leqslant minDist(q, p')$. Similarly, if we know that a peer p provides the k-th item in the answer set, then a peer p' does not need further probing if $maxDist(q, p) \leqslant minDist(q, p')$. These facts form the basis for our optimized search algorithm.

In order to formalize the above intuition, we have the following definitions.

Definition 3.1 (Dominate Relationship). *Given a query q, and two peers p_1 and p_2, if $maxDist(q, p_1) \leqslant minDist(q, p_2)$, we then say that p_1* dominates *p_2, denoted as $p_1 \prec_q p_2$, or $p_1 \prec p_2$ when q is understood.*

Theorem 3.1. *Given three peers p_1, p_2, and p_3. If $p_1 \prec p_2$ and $p_2 \prec p_3$, then $p_1 \prec p_3$.*

Proof. According to Definition 3.1, from $p_1 \prec p_2$, we know that $maxDist(q, p_1) \leqslant minDist(q, p_2)$, furthermore, from $p_2 \prec p_3$, we know that $maxDist(q, p_2) \leqslant minDist(q, p_3)$. Since $minDist(q, p_2) \leqslant maxDist(q, p_2)$, we conclude that $maxDist(q, p_1) \leqslant minDist(q, p_3)$. Therefore, $p_1 \prec p_3$. □

Definition 3.2 (Peer Domination). *Given a query q, and two groups of peers P_1 and P_2, if for each peer $p \in P_1$ and each peer $p' \in P_2$, $p \prec p'$, we say P_1 dominates P_2, denoted as $P_1 \prec P_2$.*

To graphically show the domination relationships, we plot the peers in a 2-dimensional space. Each peer is translated into one point in the 2-dimensional space, whose x-axis represents the $minDist(q, p)$ value, and the y-axis the $maxDist(q, p)$ value. Since $minDist(q, p) < maxDist(q, p)$, it is immediate that every peer (represented as a point) falls into the *triangle* region depicted

in Fig.2(a). To show the domination relationship, as depicted in Fig.2(a), the triangle region may be divided into six regions with respect to a peer p (a point in the triangle) by drawing four lines, where the two vertical lines are with Min values equal to $minDist(q,p)$ and $maxDist(q,p)$, respectively, and two horizontal lines are with Max values equal to $minDist(q,p)$ and $maxDist(q,p)$, respectively. We call such a division of the space based on the peers in the triangle region the "domination model." On the basis of these regions, we have the following observation.

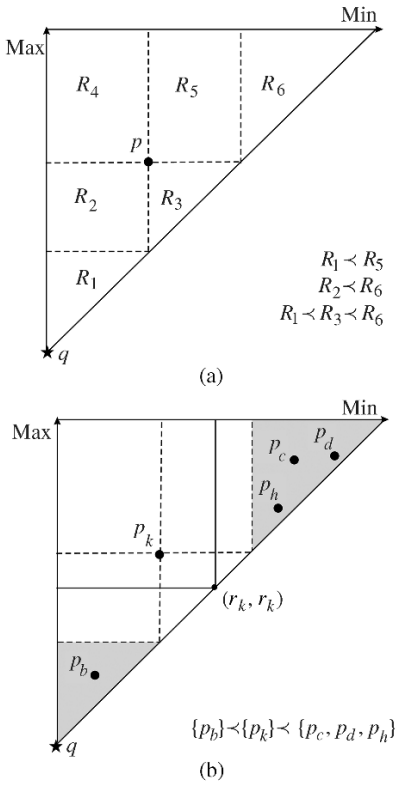


Fig.2. Domination model of query q. (a) Six regions. (b) Query radius r_k.

Observation 3.1. *For a peer p and a query q, all other peers must fall into one of the six regions divided by p. The relationships between p and peers in each region are listed as follows.*

$p.R_1$: for each peer p' in $p.R_1$, $maxDist(q,p') \leqslant minDist(q,p)$, i.e., p' dominates p.

$p.R_2$: for each peer p' in $p.R_2$, $minDist(q,p') \leqslant minDist(q,p) \leqslant maxDist(q,p') \leqslant maxDist(q,p)$.

$p.R_3$: for each peer p' in $p.R_3$, $minDist(q,p) \leqslant minDist(q,p')$ and $maxDist(q,p') \leqslant maxDist(q,p)$.

$p.R_4$: for each peer p' in $p.R_4$, $minDist(q,p') \leqslant minDist(q,p)$ and $maxDist(q,p) \leqslant maxDist(q,p')$.

$p.R_5$: for each peer p' in $p.R_5$, $minDist(q,p) \leqslant minDist(q,p') \leqslant maxDist(q,p) \leqslant maxDist(q,p')$.

$p.R_6$: for each peer p' in $p.R_6$, $maxDist(q,p) \leqslant minDist(q,p')$, i.e., p dominates p'.

Lemma 3.1. *Given a peer p, p dominates all the peers that are in region $p.R_6$, and all the peers in region $p.R_1$ dominate p.*

Proof. Given a query q, for any peer p, let min_p be the minimal distance between q and p, i.e., $min_p = minDist(q,p)$, and let max_p be the maximal distance between q and p, i.e., $max_p = maxDist(q,p)$. Using the two values min_p and max_p, we can draw six peer regions (see Fig.2(a)) w.r.t. p and q. For any peer p' in the region $p.R_6$, the minimal distance between q and p' must be larger than $maxDist(q,p)$. Therefore, $maxDist(q,p) \leqslant mindist(q,p')$. On the basis of Definition 3.1, we conclude that for any peer $p' \in p.R_6$, p dominates p', i.e., $p \prec p'$. Similarly, for any peer p'' in the region $p.R_1$, the maximal distance between q and p'' is larger than $maxDist(q,p'')$, and we conclude that for any peer $p'' \in p.R_1$, p'' dominates p, i.e., $p'' \prec p$. □

Assume we have probed a set P of peers, and found the k-th closest data elements to q among the data items contained in the peers of P. Let r_k be the distance of this k-th data, and let $p_k \in P$, called *reference peer*, be the peer providing the k-th data. Clearly, $minDist(q,p_k) \leqslant r_k \leqslant maxDist(q,p_k)$. Also, if we draw a *probe circle* centered at location of q with radius r_k in the geographical map, regions of the peers in $p_k.R_6$ do not overlap with the probe circle, since their Min values are greater than r_k. Hence, for each peer p, if $minDist(q,p) > r_k$, we can safely prune p out because it cannot contain any data lying closer to q than the current k-th data. From Fig.2(b), we can see that peers located in the right part to the vertical line $Min > r_k$ can be safely pruned. We call the rest peers *probe peers* which are those who may provide data closer to q than the current k-th data and need to be probed. Furthermore, since p_k provides the k-th current closest data, peers in its dominated region $p_k.R_6$ can be safely pruned.

4 Radius-Expansion Strategy

In order to collect answers to the k-NN query, the simplest method is to start from the query peer p_q to probe all the peers P ($p_q \notin P$) using a Steiner tree rooted at p_q. The Steiner tree is denoted by $T(p_q, P)$, or just $T(P)$ when the query peer p_q is clear from the context. The communication cost of conveying satisfying data using $T(P)$ is shown in (1).

$$C_d(P) = \beta \sum_{p_i \in P} p_i.h = \beta \cdot k \cdot |E(T(P))|, \qquad (1)$$

where $p_i.h$ is the number of data that satisfies the non-location value predicate of q, $E(T(P))$ is the set of edges in the Steiner tree $T(P)$, and $|E(T(P))|$ is the number of edges in $E(T(P))$. Obviously, the communication cost may be too high to be acceptable. In order to reduce the communication cost, we follow two directions: 1) directly probing P using as few peers as possible to get the answer; 2) collecting some statics information about and probing the "better" peers based on these statics information. For the first direction, we propose one probing strategy, namely the *radius-expansion strategy*. The radius-expansion strategy gradually expands the probe radius. For the second direction, we propose a histogram-based approach which can collect distance information about peers' data and guide the later on probing based on the histograms.

In the rest of this section, we present a filter approach for the radius-expansion probing strategy. After that, we introduce a histogram-based probing approach in Section 5.

4.1 Partition-Based Approach

The radius-expansion strategy gradually expands a probe radius. Ideally, we hope to find a small set of peers P_1, such that if P_1 can answer the k closest answers, then all the other peers can be safely pruned out. That is, if we can use our dominate diagram to classify all the peers into h successively geographical dominated groups $P_1 \prec \ldots \prec P_h$ $(h \geqslant 1)$, and the first i groups can provide k satisfying data, then the rest $h-i$ groups of peers can be pruned out. We may encounter the following two cases when we classify the peers into groups.

Case 1: each group P_i $(1 \leqslant i \leqslant h)$ contains a small number of peers, e.g., fewer than k peers, i.e., $|P_i| < k$,

Case 2: each group P_i contains a large number of peers, i.e., $|P_i| \gg k$.

To address the first case, we propose a *partition-based approach* (PA in short) that gradually extends the probing radius. For the second case, we approximately classify the peers in P_i into several "approximate" dominate groups.

With respect to Case 1, we can probe peers in P_1 first. If P_1 cannot provide k satisfying data, then it repeats the procedure by probing P_2, and so on. For each iteration, we build a Steiner tree $T(P_1)$ using the local filtering mechanism. Note that $T(P_1)$ may contain peers in the rest of group P_i, $i > 1$. Even though this approach works and is simple, when the number of dominate groups is large (an extreme case is each group contains one peer) and we need to query several groups to get the result, we have to exhaustively search in each group until k answers can be found. For example, given a source node p_q, and two groups of peers $P_1 = \{p_a, p_b\}$ and $P_2 = \{p_c, p_d\}$, Fig.3(a) shows a partial logical graph contains p_q (white node), P_1 (black node), and P_2 (grey node). Fig.3(b) is the Steiner tree of P_1. The Steiner trees of P_2 are shown in Fig.3(c). If we query $T(P_1)$ and $T(P_2)$ separately, we spend at least $5\beta \cdot k$ communication cost, whereas, as shown in Fig.3(d), querying P_1 and P_2 together (i.e., $T(P_1 \cup P_2)$) only incurs $4\beta \cdot k$ communication cost. In fact, this property has been stated and proved in the following theorem.

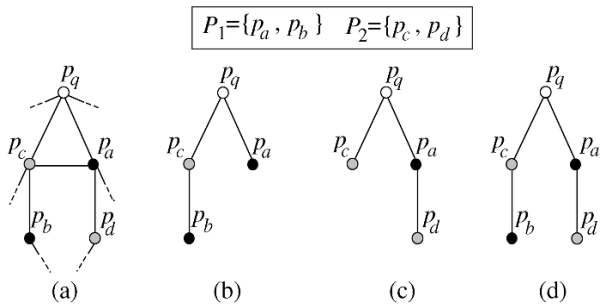


Fig.3. Pruning using exhaustive PA. (a) G. (b) $T(P_1)$. (c) $T(P_2)$. (d) $T(P_1 \cup P_2)$.

Theorem 4.1. *Given an undirected graph $G(P, E)$ and two sets of target nodes $P_1 \subseteq P$ and $P_2 \subseteq P$. Let $p_q \in P$ be a query peer, we have the following property:*

$$C_d(P_1) + C_d(P_2) \geqslant C_d(P_1 \cup P_2).$$

Proof. According to (1), we prove $|E(T(p_q, P_1))| + |E(T(p_q, P_2))| \geqslant |E(T(p_q, P_1 \cup P_2))|$. On the basis of the undirected graph $G(P, E)$, we constructed a Steiner tree $T(p_q, P_1)$. Let N_1 be Steiner vertices in $T(p_q, P_1)$. The number of edges in $T(p_q, P_1)$ is the number of nodes in $T(p_q, P_1)$ minus 1, i.e., $|E(T(p_q, P_1))| = |\{p_q\} \cup P_1 \cup N_1| - 1$. According to the definition of Steiner tree, $N_1 \cap \{\{p_q\} \cup P_1\} = \emptyset$. In our context, $\{p_q\} \cap P_1 = \emptyset$, therefore, $|E(T(p_q, P_1))| = |P_1| + |N_1|$. For the same reason, let N_2 and N be Steiner vertices in $T(p_q, P_2)$ and $T(p_q, P_1 \cup P_2)$, respectively. We know $|E(T(p_q, P_2))| = |P_2| + |N_2|$ and $|E(T(p_q, P_1 \cup P_2))| = |P_1 \cup P_2| + |N|$.

Suppose the above property does not hold, then let $|E(T(P_1))| + |E(T(P_2))| < |E(T(P_1 \cup P_2))|$. Now we prove that this statement does not hold.

According to the above analysis, we rewrite the above statement into $|P_1| + |N_1| + |P_2| + |N_2| < |P_1 \cup P_2| + |N|$. Since P_1 and P_2 are the two sets of target peers, we know $|P_1| + |P_2| \geqslant |P_1 \cup P_2|$ must hold. Therefore, if the above statement holds, then $|N_1| + |N_2| < |N|$ should hold. That is, we can find a peer p' in N, such that $p' \notin N_1$, $p' \notin N_2$.

Since p' is a Steiner vertex in the Steiner tree $T(p_q, P_1 \cup P_2)$, let $P' \in P_1 \cup P_2$ be the set of peers that are reached by p_q using p' in $T(p_q, P_1 \cup P_2)$. We

can use the peers in N_1 and N_2 to replace p', so that P' can still be reached by p_q using other Steiner vertices in N_1 and N_2. We keep removing those peers that only in N and add peers in N_1 and N_2 to make $P_1 \cup P_2$ reachable by p_q. Finally, we can construct a new tree T' rooted at p_q that contains all the peers in P_1 and P_2. According to the definition of Steiner tree, $T(p_q, P_1 \cup P_2)$ should be the tree with the smallest edges, so we cannot construct such T'. Therefore, the assumption $|E(T(P_1))|+|E(T(P_2))|<|E(T(P_1 \cup P_2))|$ does not hold. We conclude $|E(T(p_q, P_1))| + |E(T(p_q, P_2))| \geqslant |E(T(p_q, P_1 \cup P_2))|$, that is, $C_d(P_1) + C_d(P_2) \geqslant C_d(P_1 \cup P_2)$ holds. □

The PA approach tries to find and merge groups (such as P_1 and P_2) to save more communication cost. The basic task for the PA is the following.

Definition 4.1 (Optimal Peer-Partition). *Given an undirected graph $G = (V, E)$, a set of terminal nodes $R \subseteq V$, and subsets $R_1, \ldots, R_h$, $R_i \subseteq R$ $(1 \leqslant i \leqslant h)$, find an optimal partition $P' = \{P_1, P_2\}$, such that neither P_1 nor P_2 is empty, for each R_i, it can only belong to P_1 or P_2, and the sum of the numbers of edges in the Steiner trees for P' is minimal.*

Theorem 4.2. *The problem of optimal peer-partition is NP-hard.*

Proof. We prove it by giving a reduction from the *partition* problem to optimal peer-partition problem in polynomial time.

Partition Problem: given a set S of n integers, is it possible to partition S into two subsets S_1 and S_2 so that the sum of the integers in S_1 is equal to the sum of the integers in S_2?

Given a set S of n integers, we sort these n integers in ascending order $\{I_1, \ldots, I_n\}$. We construct a tree $T(v)$, where v is the root node of T with level 0. We construct I_1 children of v, i.e., $\{v_1, \ldots, v_{I_1}\}$. For each node in the i-th level $(1 \leqslant i \leqslant n)$, we construct I_{i+1} children. So, the constructed tree has $n + 1$ levels and contains $1 + \sum_{j=1}^{n} \prod_{i=1}^{j} I_i$ nodes. Clearly, this construction can be done in polynomial time. We use four integers {1, 2, 3, 4} to illustrate the construction of the tree, shown in Fig.4. In the constructed tree, s is the root node, the tree level is 5 and there are 34 nodes.

We show the above two problems are equivalent by proving the following. We define the root node as *source node*. For any subset $S' \subset S$, we rank elements in S' in ascending order $\{I_1, \ldots, I_m\}(m < n)$. We define *target nodes* by choosing $\prod_{i=1}^{j} I_i$ nodes in the j-th level. For example, given a subset {2,3}, target nodes consist of 2 nodes in the 2nd level and 2×3 nodes in the 3rd level (see Fig.4(b), where the black nodes are chosen to be target nodes). We show that if every one of the nodes in j-th level has the same parent, and for any node in the h-th level $(j > h)$ the chosen nodes are ancestors of the chosen nodes in j-th level, then constructing Steiner tree to reach chosen nodes is minimum. Therefore, given the two sets {1, 4} and {2, 3} with the same summation, we can construct two Steiner trees such that their edges are minimum.

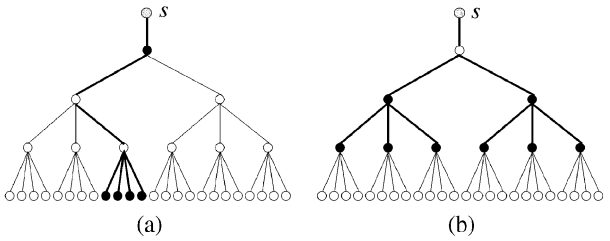


Fig.4. Example of the constructed tree and their minimal partition trees. (a) Steiner tree for $\{1, 4\}$. (b) Steiner tree for {2, 3}.

In summary, the problem of choosing a partition $S_1 = S_2$ in S is equivalent to calculating two Steiner trees to make the summation of their edges minimum. Thus our optimal peer-partition problem is NP-hard. □

We use a greedy algorithm to approximately build two Steiner trees, which is described in Algorithm 1. PA combines geographical dominate relationship and logical Steiner tree together. Given a set of dominate groups of peers $P_1 \prec \ldots \prec P_h$ $(1 \leqslant i \leqslant h)$, we first build up a Steiner tree for a source node p_q and each terminal node set P_i $(1 \leqslant i \leqslant k)$ for the first k groups using the

Algorithm 1. PA

Input: k, undirected graph G, query peer p_q,
peer sets $P' = \{P_1, \ldots, P_h\}$, where $P_1 \prec \cdots \prec P_h$;
Output: two Steiner trees T_1 and T_2;

```
1:  for (each P_i between P_1 and P_k) do
2:      ST_i = greedyST(G, p_q, P_i);
3:  end for
4:  find two minimal trees ST_u and ST_v corresponding
    to P_u and P_v;
5:  T_1 = ST_u;    S_1 = P_u;
6:  T_2 = ST_v;    S_2 = P_v;
7:  for (each P_i in P' − S_1 − S_2) do
8:      pick a minimal tree ST_i for P_i;
9:      T'_1 = greedyST(T_1 ∪ T_i, p_q, S_1 ∪ P_i);
10:     T'_2 = greedyST(T_2 ∪ T_i, p_q, S_2 ∪ P_i);
11:     if (|E(T'_1)| ⩽ |E(T'_2)|) then
12:         T_1 = T'_1;    S_1 = S_1 ∪ P_i;
13:     else
14:         T_2 = T'_2; S_2 = S_2 ∪ P_i;
15:     end if
16: end for
17: return T_1 and T_2;
```

greedy Steiner approach[14]. The function greedyST, shown in line 2, contains three parameters, the first is an input graph $G = (V, E)$, the second is the source node $p_q \in V$, and the last is a set of terminal nodes. $V' \subseteq V$. The algorithm PA greedily expends a subtree T of G that includes p_q and V' by inserting a shortest path from node in G to T. Lines 4∼6 describe the initial partition by choosing two Steiner trees with the minimal edges as the seeds of a partition. Lines 7∼16 greedily insert the remaining Steiner trees into the two seeds and compare the numbers of edges for each possible partition S_1 and S_2. A partition with minimal edges will be kept. We repeat this procedure until all the remaining Steiner trees have been processed.

PA classifies some groups of peers into a class C_{P_1} so that these groups can be probed together. Note that, groups in $C_{P_2} = C_P - C_{P_1}$ may contain a group P_j dominates groups in C_{P_1}. For example, given $P_1 \prec P_2 \prec P_3 \prec P_4$, we can get $C_{P_1} = \{P_1, P_3\}$ and $C_{P_2} = \{P_2, P_4\}$. If the first round probe gets k' ($\leqslant k$) closest answers, then using the locations of these k' answers, we can know k_1 ($\leqslant k'$) out of k' answers is returned by P_1. Then it classifies $P' - C_{P_1}$ into two classes $C_{P_1} = \{P_2\}$ and $C_{P_2} = \{P_4\}$ and request $k - k_1$ to C_{P_1}. PA iterative repeats the above procedures until k closest answers are returned.

The time complexity of greedyST(G, p_q, P_i) is $O(mn^2)$, where m is the number of nodes in $p_q \cup P_i$, and n is the number of nodes in G. For each peer set P_i, the average size of P_i is $\frac{n}{h}$, Algorithm 1 processes k peer sets, therefore, the time complexity of Algorithm 1 is $O(\frac{k}{h}n^3)$.

4.2 Preprocessing for PA-Based Approaches

As mentioned in Subsection 4.1, we may encounter the Case 2 that the number of peers in a group (after we classify peers into h successively geographical dominate groups) is large. It is not efficient to probe all the peers in such a large group. In this case, we seek to break such a group into small approximate dominate sets, so that radius-expansion strategy can be employed to efficiently prune out the peers.

Fig.5 describes how to approximately classify the peers. For each radius r, there is a corresponding point $S = (r, r)$ in Fig.5, which divides the peers into three regions R_1, R_2, and R_3. We know that the peers in region R_1 dominate all the peers in region R_3. For each peer p in R_2, $minDist(q, p) \leqslant r \leqslant maxDist(q, p)$. We call the peers in R_2 *outlier peers to* R_1. Ideally, the number of outlier peers to R_1 equals zero, then the radius r can classify the peers in Fig.5 into two dominate groups corresponding to R_1 and R_3, respectively. However, in the case that we cannot find a radius r such that its corresponding R_2 contains zero outlier peers, we seek the radius whose corresponding R_2 regions contains a small number of outlier peers (i.e., less than τ% of the number of peers, where τ is a threshold). If the number of outlier peers in R_2 is less than $\tau \times n$ (n is the number of peers), then the corresponding radius is called *separable radius*. We gradually increase r from 0 to r_k (the current probing radius). For the first separable radius r_1, we classify all the peers in region $r_1.R_1$ ($r_i.R_j$ refers to a region R_j with r_i as separable radius) into a group P_1, and all the peers in region $r_1.R_2$ in the outlier set of P_1 (denoted as O_1). We then use the second separable radius r_2 to classify peers in $r_1.R_3$ into P_2 and O_2, respectively. We repeat this procedure, until all the peers have been grouped to either dominate groups or different outlier groups.

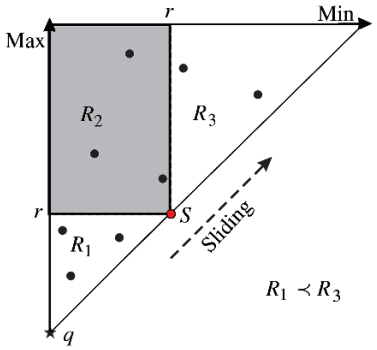


Fig.5. Approximately classifying dominate sets.

5 Histogram for Optimal Probing Peers

When we probe m groups of peers $\{P_1, \ldots, P_m\}$, separately, to get k data satisfying the query from each group, the total communication cost is

$$\sum_{j=1}^{m} C_d(P_j). \tag{2}$$

However, if we can merge all the probing peers into one group and submit query using one Steiner tree to cover m groups, denoted as $T(\cup_{j=1}^{m} P_j)$, the communication cost may be further reduced (see Theorem 4.1). We use (3) to describe it.

$$\sum_{j=1}^{m} C_d(P_j) \geqslant C_d\Big(\bigcup_{j=1}^{m} P_j\Big). \tag{3}$$

The left part of the inequation expresses the communication costs of probing peers in m separated Steiner trees, each of which covers the peers in a group P_i, and the right of the inequation expresses the communication

costs of one large Steiner tree $T(\bigcup_{j=1}^{m} P_j)$ that covers peers in all these m groups. Obviously, forwarding values in the Steiner tree $T(\bigcup_{j=1}^{m} P_j)$ requires less communication cost than the summation costs of forwarding values in m Steiner trees $T(P_1), \ldots, T(P_m)$, respectively.

The question here is how can we determine a "good" m? On one hand, if m is small, then we have to probe the peers for more times to get all the candidates. On the other hand, if m is too large, then the filterability will be decreased.

In fact, besides directly probing the peers, we can first collect some statistic data about minimum and maximum distances and carry out the probing later on based on the collected statistic information. Obviously, collecting statistic data needs some communication cost which seems contradict to our previous sought, reducing the communication cost as much as possible. However, with the help of the statistic information, we can select much "better" peers to probe and to save more communication cost as a consequence. In other words, we are seeking a solution which can gain more communication savings overall. Moreover, as we want to propose a solution to address the continuous queries which adapt to the dynamic changes over the P2P system, collecting statistic information is a must to incrementally build the results based on previous answers, which will be discussed in detail in the next section. Therefore, from the whole system solution point of view (including answering query and adjusting according to the updates), collecting statistic information does not incur any extra communication cost, since we have to do it anyway to adapt to the dynamic environment.

In order that with less communication cost to get to know whether the first m groups of peers can return k data, we propose a histogram-based approach to optimal probing peers, specifically, the query peer p_q that creates a histogram to count the number of satisfying data in each bucket. In order to create such a histogram, we have to collect the data information of the peers. The communication cost for gathering histograms from m individual groups of the peers is shown in (4).

$$C_H = N_b \cdot \gamma \sum_{j=1}^{m} |E(T(P_j))|, \tag{4}$$

where γ is the size of an integer to represent a bucket count, N_b is the number of buckets and each bucket is associated with a distance interval in $[0, r_k)$. With the help of this histogram, we can find a least query radius r_a (called *estimated query radius*) such that the peers within r_a whose minimal distances to q less than r_a can get k-*NN* results.

As mentioned before, we want to achieve the goal that the cost of gathering histogram together with later on probing is less than the communication cost of direct probing. We do this through carefully choosing N_b. Let N_{b_m} be the *maximal number of buckets* such that the constraint in (5) must be satisfied.

$$\sum_{j=1}^{m} C_d(P_i) \geqslant C_H + C_d\Big(\bigcup_{j=1}^{m-1} P_j \cup P'_m\Big), \tag{5}$$

where P'_m is a subset of P_m, whose minimal distances are less than the estimated query radius r_a, which is calculated using gathered histogram. We can use P_m to replace P'_m, and get a smaller number $N'_{b_m} \leqslant N_{b_m}$ shown in (6).

$$N'_{b_m} \leqslant \left\lfloor \frac{\sum_{j=1}^{m} C_d(P_i) - C_d\Big(\bigcup_{j=1}^{m} P_j\Big)}{\gamma \sum_{j=1}^{m} |E(T(P_j))|} \right\rfloor. \tag{6}$$

Algorithm 2. Construct Histogram with Adaptive Buckets

Input: query q, k dominate groups $P = \{P_1, \ldots, P_k\}$ such that for any $p \in P$, $minDist(q, p) \leqslant r_k$;

Output: buckets for P;

```
1:  let L be an empty set;
2:  for each group P_i (1 ⩽ i ⩽ k) do
3:      let L' be an empty set;
4:      for each p ∈ P_i, insert minDist(q,p) and
        maxDist(q,p) into L';
5:      sort L' in ascending order;
6:      L = L ∪ L';
7:  end for
8:  for any two adjacent elements l_j, l_{j+1} ∈ L do
9:      construct a peer interval [l_j, l_{j+1});
10: end for
11: calculate the maximal N'_{b_m} for group P
12: h = number of elements in L;
13: probe_interval = [L[0], L[h − 1]);
14: if N'_{b_m} equals to 0 then
15:     return ∅;
16: else if N'_{b_m} ⩾ h then
17:     build a bucket for each interval [l_j, l_{j+1})
        ∈ [L[0], L[h − 1]);
18: else
19:     build one bucket for the first h − N'_{b_m} peers
        intervals in [L[0], L[h − 1]);
20:     build one bucket for each remaining peer
        intervals in [L[0], L[h − 1]);
21: end if
22: return buckets;
```

Algorithm 2 describes the steps to construct a histogram with adaptive buckets. Given peers overlapping with the probe radius r_k. Lines 2∼7 sort $minDist(q,p)$ and $maxDist(q,p)$ of every peer p in ascending order. Let $L = \{l_1, \ldots, l_m\}$ be this sorted list, where l_i $(1 \leqslant l_i \leqslant m)$ is either $minDist(q,p)$ or $maxDist(q,p)$. We call each l_i *distant point*. Lines 8∼10 produce a set of *peer intervals* $[l_1, l_2), \ldots$, and $[l_{m-1}, l_m)$ using distant points.

For each probe, we can calculate the maximal possible bucket number. In the m-th probe, assume all peers' minimal distances and maximal distances belong to the *group interval* $[r_{m-1}, r_m)$. We probe peers P_m overlapping with the interval $[r_{m-1}, r_m)$. If N'_{b_m} is large enough, then we build a bucket for each peer interval $[l_i, l_{i+1}) \in [r_{m-1}, r_m)$ $(1 \leqslant i < m)$. Such a fine granularity histogram can provide more accurate estimated query radius. However, a larger number of buckets lead to more communication cost. Therefore, we classify several peer intervals $[r_{m-1}, l_1), [l_1, l_2), \ldots, [l_u, r_m)$ into N'_{b_m} buckets.

If N'_{b_m} equals 0 (Line 14), peers in P_m directly convey their satisfying data to the query peer p_q during the probe phases, so that the minimal communication cost constraint can be hold. Furthermore, the query peer p_q maintains a *global histogram* by building one bucket for the group interval $[r_{m-1}, r_m)$.

If N'_{b_m} is larger than the number of generated intervals h (Line 16), each probe peer conveys its histogram to the query peer p_q. p_q collects these local histograms from all probed peers and merge them into the global histogram. p_q determines an estimated query radius r_a by choosing the first minimum h buckets $(h < N'_{b_m})$ whose number of answers can provide k answers. Then p_q queries peers overlapping with $[0, r_a)$ to get result.

Assume there are h peer intervals $[r_{m-1}, l_{u+1}), [l_{u+1}, l_{u+2}), \ldots, [l_{u+h-1}, r_m)$ in $[r_{m-1}, r_m)$. If $h \leqslant N'_{b_m}$, then we construct h buckets, each of which corresponds to a peer interval. During the probe phases, each peer in group P_m counts satisfying data for each bucket and conveys its local histogram to p_q.

In the case that h is larger than N'_{b_m}, we build fine granularity buckets for the last few peer intervals near to r_m so that the last few buckets can record more accurate statistical information, which result in high probability to filter a small number of peers whose minimal and maximal distances to q are near to r_m. Therefore, for each peer p, we associate $N'_{b_m} - 1$ buckets with the last $N_{b_m} - 1$ peer intervals, respectively. We associate the first bucket with the rest intervals. The time complexity of Algorithm 2 is linear with the number of peers in P, i.e., $O(|P|)$.

For example, Fig.6(a) shows three groups of peers P_1, P_2, and P_3 (marked using different colors, respectively) associated with the query interval $[r_5, r_6)$, $[r_6, r_7)$, and $[r_7, r_8)$. Assume for the first group of peers P_1, there are six peer intervals $[r_5, l_1)$, $[l_1, l_2)$, $[l_2, l_3)$, $[l_3, l_4)$, $[l_4, l_5)$, $[l_5, r_6)$, and the calculated $N'_{b_m} = 4$. We use one bucket to associate the first three peer intervals $[r_5, l_1)$, $[l_1, l_2)$, and $[l_2, l_3)$, and the other three buckets to associate the rest three peer intervals, respectively (shown in Figs.6(b), 6(c), and 6(d)).

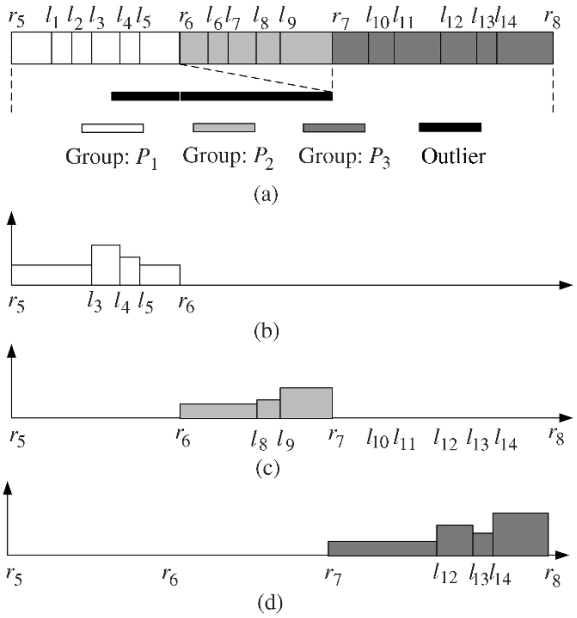


Fig.6. Histogram of four groups of peers. (a) Peer groups and the query intervals $[r_5, r_6)$, $[r_6, r_7)$, and $[r_7, r_8)$. (b) Local histogram for the peer group P_1. (c) Local histogram for the peer group P_2. (d) Local histogram for the peer group P_3.

For simply maintaining global histogram, p_q does not collect histogram of outlier peers during the probe phase. For example, no local histogram count the number of satisfying data in the outlier peer (expressed as black box) in Fig.6. During the query phase, using the calculated estimated query radius r_a, an outlier peer whose minimal distance less than r_a should be added for query processing. Using the returned satisfying data during query phase, p_q can classify data of outlier peers into different buckets of the global histogram to make it more accurate.

Furthermore, as discussed in Section 6, we utilize these computations for the efficient handling of updates.

6 Continuous *k*-*NN* Queries

Continuous k-NN queries are issued once and long-running continuously to generate k closest results along with the updates of the peers. In this section, our task is

to maintain the results of the queries upon the arrival of peers or their data updates. Therefore, we first analyze behaviors of peers and their data. We then propose algorithms for the efficient handling of updates using the constructed histogram in the above section.

6.1 Updates of Peers and Data

For a cooperative peer in an unstructured P2P system, when a peer or its managed data changes, it has responsibility to inform the whole system about its update, so that the system can make quickly response to the update. Now we analyze the peer updates in an unstructured P2P system.

As we know, a peer in a P2P system can freely join or leave in the system momentarily, and their cooperative relationship among peers can also be changed dynamically. We consider the change of cooperative relationships as a peer joins and leaves. A peer's join or leaving from the system can be expressed as a node insertion or deletion from the dominate model. As shown in Fig.7(a), a peer p_a leaves and a new peer p_h joins. Furthermore, a peer region can also expand or shrink dynamically, since its managed spatial data may change its shape.

Lemma 6.1. *Given a query q, if a peer p expands to p', then $minDist(q,p') \leqslant minDist(q,p)$ and $maxDist(q,p') \geqslant maxDist(q,p)$. If a peer p shrinks to p', then $minDist(q,p') \geqslant minDist(q,p)$ and $maxdist(q,p') \leqslant maxDist(q,p)$.*

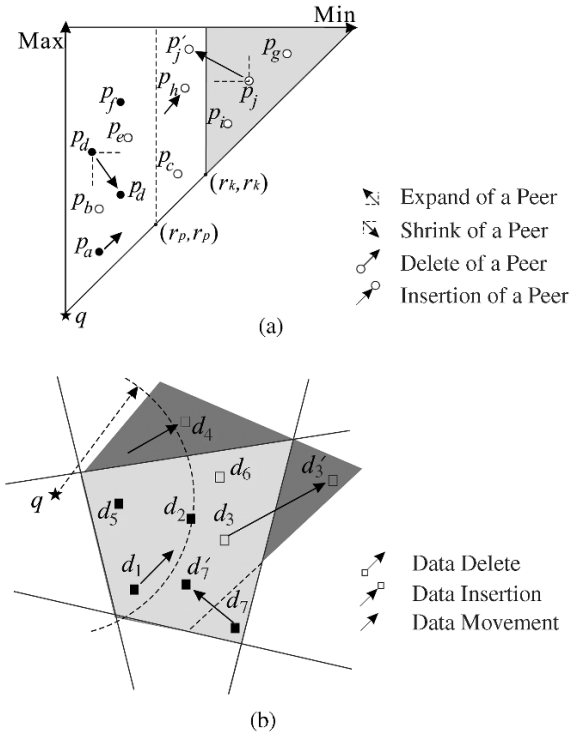


Fig.7. Updates of autonomous peers. (Black nodes and boxes represent the peers and data satisfying query predicates, respectively.) (a) Peer updates. (b) Data updates.

Proof. According to the definition of spatial region $R(p)$ of a peer p, when p expands to p', there must exist data d, whose value of location attribute exceeds the spatial region $R(p)$ (convex hull) of data in $D(p)$. That is, the location value of d, $\boldsymbol{v}_l(d)$, is a new convex to expand the convex hull from $R(p)$ to $R(p')$. The distances $minDist(q,p)$ and $maxDist(q,p)$ could be changed due to the insertion of d. If $\boldsymbol{v}_l(d)$ is in between q and $minDist(q,p)$, then $minDist(q,p') < minDist(q,p)$. If $\boldsymbol{v}_l(d)$ is outside $maxDist(q,p)$, then $maxDist(q,p') > maxDist(q,p)$. For other cases, $minDist(q,p')$ and $maxDist(q,p')$ do not change.

When removing a convex d from $D(p)$, the spatial region $R(p)$ for the peer p will shrink to $R(p')$. If d is the only one convex that determines $minDist(q,p)$ in $R(p)$, then $minDist(q,p) < minDist(q,p')$. If d is the only one convex that determines $maxDist(q,p)$ in $R(p)$, then $maxDist(q,p) > maxDist(q,p')$. For the remaining cases, $minDist(q,p')$ and $maxDist(q,p')$ do not change. □

According to Lemma 6.1, if a peer expands, its corresponding node in the dominate diagram either moves to its left upper corner or does not move, and if a peer shrinks, its corresponding node either moves to its right lower corner or does not move. For example, p_j expands its region to p'_j and p_d shrinks its region to p'_d. Since a peer region can be changed to any shape, simply, we regard this change as an expand plus a shrink.

Besides peer update, data in each peer is also changed dynamically. Fig.7(b) shows three kinds of data update. (i) Data is inserted into or deleted from a peer, e.g., data d_1 is removed from the peer, whereas d_4 is inserted. Note that, data insertion or deletion may cause the shape change of the corresponding peer (see the dark grey area). (ii) Location of data changes, e.g., d_3 moves to d'_3 and d_7 moves to d'_7. A movement of data can be treated as delete and insertion of data. (iii) Value of data changes, e.g., the temperature detected by a sensor changes from 40 degree to 90 degree. We can regard this kind of update as deletion of a sensor of 40 degree and insertion of a sensor of 90 degree, i.e., data value change can be treated as data deletion and insertion with unknown data value. In summary, updates (i) and (ii) may cause the change of a peer's shape. For any update data d' in a peer p, if $dist(q,d') < minDist(q,p)$ or $dist(q,d') > maxDist(q,p)$, then it causes p to expand or shrink.

Table 2 summarizes the above updates of a peer p' and its data d'. We only show the cases where

p' is changed within the estimated query radius r_a, i.e., $minDist(q,p') < r_a$, because update outside of r_a will not affect the *k-NN* result. In Cases 1) and 2), a peer p' joins into the system or p' expands its region. In Case 1), update of p' affects *k-NN* result ($minDist(q,p') < r_p$), i.e., we need to decrease r_p to a new perfect query radius r'_p, if p' can provide at least one satisfying data. In Case 2), update of p' does not affect the *k-NN* result, since $dist(q,d) \geqslant r_p$.

Table 2. Update of a Peer p' ($minDist(q,p') < r_a$)

Type	Condition	Case
Peer Insertion/Expand	$minDist(q,p') < r_p$	1)
	$minDist(q,p') \geqslant r_p$	2)
Peer Delete/Shrink	p' provides answer	3)
	else	4)
Data Update	$minDist(q,p') < r_p$	5)
	$minDist(q,p') \geqslant r_p$	6)

In Cases 3) and 4), a peer p' leaves from the system or p' shrinks its region. In Case 3), if a leaving peer p' provides m ($< k$) answers in the *k-NN* result, then we need to find other peers to provide m closest answers. If a peer p shrinks to p', i.e., $minDist(q,p') \geqslant minDist(q,p)$ and $maxDist(q,p') \leqslant maxDist(q,p)$ due to some data delete or movement, then the previous *k-NN* result may not provide enough answers. The query radius r_p should be extended to a larger r'_p so that the peers within r'_p can provide enough answers. Case 4) will not change answers to the query since p' does not provide answers to q.

In Cases 5) and 6), when data in a peer p' changes, p' notifies the system that its data has changed so that the system can correspondingly reflect. In Case 5), data is inserted or deleted. We treat p' as a new insertion peer shown in Case 1). Similar with Case 2), update of data in p' cannot affect the result since $minDist(q,p') \geqslant r_p$.

6.2 Efficiently Query Using Histogram

Based on the analysis in the above subsection, we classify the reaction of the query peer p_q into two categories, which are *increasing* the result radius r_p and *decreasing* the result radius r_p.

6.2.1 Increasing the Result Radius r_p

In Case 3), when a peer p is deleted or its region shrinks, we need to extend r_p to get enough answers to the *k-NN* query q. Assume that p provides h ($\leqslant k$) answers, then the result radius r_p has to be increased to probe more peers to get another h closest answers. Algorithm 3 describes the procedure of maintaining and using the global histogram to incrementally query a small set of peers.

Algorithm 3. Increasing r_p

Input: global histogram, a delete peer p that provides h answers, answer set $ASet$;

Output: the global histogram after update of p;

```
1:  find distant points L equal to minDist(q,p) or maxDist(q,p);
2:  for each distant point l_i ∈ L do
3:      if l_i is only associated with p then
4:          remove l_i from the distant points set in the histogram;
5:          merge buckets [l_{i-1}, l_i) and [l_i, l_{i+1}) into one bucket [l_{i-1}, l_{i+1});
6:      end if
7:  end for
8:  find minimum closest buckets B to answer h data;
9:  query q to peers and outliers associated with B;
10: insert answers to ASet;
11: choose k closest data from ASet;
12: r'_p = distance between q and the k-th data;
13: return global histogram;
```

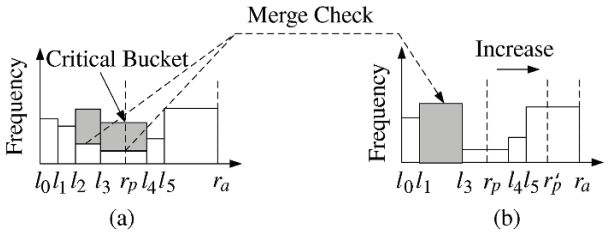


Fig.8. Adjusting global histogram by merging buckets and increasing r_p to r'_p. (a) Global histogram before deleting peer p. (b) Global histogram after deleting peer p.

To update a peer p may cause some distant radiuses disappear. Line 1 finds distant points who are equivalent to $minDist(q,p)$ or $maxDist(q,p)$ in the global histogram. For example, Fig.8(a) shows a global histogram, which contains six distant points $\{l_0,\ldots,l_5\}$. A removed peer p has two distant radiuses $minDist(q,p) = l_2$ and $maxDist(q,p) = l'$ ($l_3 \leqslant l' \leqslant l_4$). l_2 is a distant point that equals $minDist(q,p)$. In Algorithm 3, Lines 2∼7 adjust buckets in the global histogram. For instance, there are two buckets with intervals $[l_2,l_3)$ and $[l_3,l_4)$ that are affected and should be adjusted when p is removed from the system. Assume l_2 is only associated with p, then l_2 is no longer a distant point after deleting p. We need to merge $[l_1,l_2)$ and $[l_2,l_3)$ into one bucket $[l_1,l_3)$ (shown in Fig.8(b)).

In Fig.8, r_p is the result radius of the current *k-NN* result and r_a is the estimated query radius. Let h be the *invalid answer* (represented as grey color) of p within r_p. We then get a new global histogram (without grey part in Fig.8(a)) by removing invalid answers in the bucket $[l_1,l_3)$.

Line 8 determines how many additional peers within the estimated query radius r_a can provide the missing h closest answers to q. We then extend r_p to r'_p. We find minimal additional buckets that can provide at least h data in the global histogram, i.e.,

$$\text{minimize } i, \quad \text{s.t.} \sum_{i=B_c}^{N} freq(i) \geqslant h, \tag{7}$$

where N is the number of buckets in the global histogram, B_c is the bucket intersect with r_p (called *critical bucket*). Lines 9∼11 find associated peers and outliers with the calculated buckets, and submit q to those peers to get h satisfying data. Then the query peer p_q chooses k closest data from the exiting answer set $ASet$ and Line 12 changes radius from r_p to r'_p.

The time complexity of Algorithm 3 is $O(n)$, where n is the number of buckets in the global histogram. Notice that, if the frequency summation of the buckets between the critical bucket and r_a is less than h, i.e., the global histogram has no enough statistical information within r_a, then r_p need re-compute peers in r_k and get the new h closest answers.

6.2.2 Decreasing the Result Radius r_p

In Cases 1) or 5), the update of a peer p may cause decreasing r_p to r'_p. The query peer p_q first submits q to p along the shortest path from p_q ro p to get satisfying data whose distance to q less than r_p. Then p_q chooses k closest data from the current answer set and the new returned data.

In order to incrementally probe affected peers in Case 3), p_q adjusts the global histogram using the returned data of p. For example, the grey part in Fig.9(a) is the increased number of answers in the buckets $[l_1, l_2)$ and $[l_2, l_3)$.

When the number of answers in one bucket is larger than a specified number (e.g., k), we split this bucket into two buckets. In Fig.9(a), the frequency of bucket $[l_1, l_2)$ is $H(>k)$, so we split buckets as described in Algorithm 4. For an insert or expand peer p, Lines 1∼2 calculate new distant radius using p. Lines 4∼14 choose a separate radius to split a bucket into two buckets. For example, if $minDist(q,p) = l'$ and $maxDist(q,p) = l_3$, then l' is a new distant radius. We use l' as a separate point to split bucket $[l_1, l_2)$ into two buckets $[l_1, l')$ and $[l', l_2)$. Let h_1 be the number of satisfying data locates in $[l_1, l')$ and h_2 be the number of satisfying data locates in $[l', l_2)$. Then the frequency of the bucket $[l_1, l')$ is between h_1 and h_1+H, and the frequency of the bucket $[l', l_2)$ is between h_2 and h_2+H. The global histogram records such frequency interval instead of the accurate frequencies for split buckets.

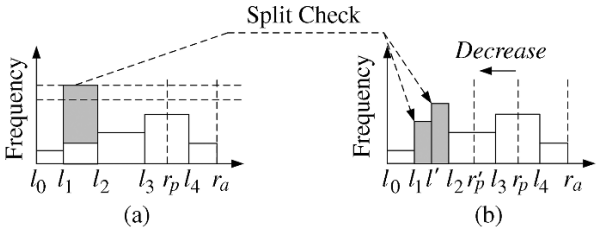


Fig.9. Adjusting global histogram by splitting buckets and decreasing r'_p to r_p. (a) Global histogram before updating peer p. (b) Global histogram after updating peer p.

Algorithm 4. Lazy Split Buckets

Input: an insert or expand peer p, H value, a set of satisfying data $D = \{d_1, \ldots, d_n\}$ from p, $dist(q, d_i) < r_k$;

Output: two split buckets B_1 and B_2;

1: distant radius $r_1 = minDist(q, p)$;
2: distant radius $r_2 = maxDist(q, p)$;
3: find bucket B that crosses r_1 (or r_2, resp.);
4: **if** fequency of B is larger than H **then**
5: **if** r_1 or r_2 is a new distant radius **then**
6: random choose r_1 or r_2 as the separate radius l';
7: **else** $l' = \frac{1}{2}(B.min + B.max)$;
8: **end if**
9: split B into B_1 and B_2 into $[B.min, l')$ and $[l', B.max)$;
10: $h_1 = |D_1|$, $D_1 \subseteq D$ and $\forall d \in D_1$, $dist(q, d) \leqslant l'$;
11: $h_2 = |D_2|$, $D_2 \subseteq D$ and $\forall d \in D_2$, $dist(q, d) > l'$;
12: let $freq(B_1)$ be $[h_1, h_1 + H]$;
13: let $freq(B_2)$ be $[h_2, h_2 + H]$;
14: **end if**
15: **return** B_1 and B_2;

Note that, the global histogram maintains a frequency interval for the split buckets. We do not calculate the precise frequency since we have no clue of the data distribute in the bucket B. We call this procedure *lazy split*. On the basis of this interval, we can estimate a set of affected peers when a peer leaves or its region shrinks. Those affected peers should be probed and precise frequencies of the associated buckets can be got.

7 Related Work

As far as we know there is no similar work on the proposed problem setting. The work most closed to ours falls into two categories: continuous *k-NN* searches over spatial database and *k-NN* queries over structured P2P systems. We will review the work in these two directions and explain the difference between our work and theirs. There are many methods that have been proposed for

continuous *k-NN* queries over moving objects in spatial database domain. Most of them focus on reducing the number of updates to the indexes. In order to achieve this, the trajectories of moving objects are modelled by some linear functions, thus an R-tree can be built using time as a function[15,16]. Compared to these work, our problem setting is for distributed environment and we do not assume the existence of a centralized index. Moreover, we have the logical communication cost as a constraint to the *k-NN* queries.

k-NN queries over P2P systems can be classified into search over unstructured P2P and structured P2P. For structured P2P systems, data allocation strategies are important for *k-NN* search. Distributed Hashing Table (DHT) is often used to allocate data, such as CAN[17], Chord[18], Pastry[19], and Tapestry[20], which use uniform hash functions and achieve good load balance. However, these hashing functions destroy data locality (data that are similar should be allocated near to each other in the space). Complicated queries such as *k-NN* have to rely on multi-cast or additional indexes. Some locality-preserving data allocation approaches are also proposed, such as P-Grid[21], P-Ring[22], Baton[11], Vbi-tree[10], and Mercury[23]. The basic idea of these approaches is to keep data locality over the attribute as much as possible. For unstructured P2P systems, very little work has been done so far. Gnutella[24] uses flooding techniques to do *k-NN* search. Compared to the above work, our work focuses on continuous constrained *k-NN* search over unstructured P2P system. Besides finding the *k-NN*, we have to guarantee that searched value satisfies the value predicate specified in the query. Furthermore, in contrast to the traditional setting of P2P systems that each peer maintains static data, a peer in our system maintains dynamic data.

Basically, the research in this area can be classified into two categories: indexing structures and searching algorithms. In the first category, many indexing structures have been proposed to speed up the retrieval efficiency, such as R-tree[25], K-d tree[26], TV-tree[27], SR-tree[28], and X-tree[29]. If the distance function *dist* on $\mathcal{D}$ is a metric distance function, indexing structures on metric space can be applied, such as GH-tree[30], GNAT[31], SA-tree[32], M-tree[33], and MVP-tree[34]. In the second category, many efficient search algorithms have been proposed to reduce the number of disk page access during a *k-NN* search[1,2,4]. One common aspect of the above indexing structures and searching algorithms is that they work on a single (central) data set. On the contrary, our work on *k-NN* search focuses on a distributed environment, specifically, in an unstructured P2P system. Furthermore, in contrast to the traditional setting of P2P systems that each peer maintains static data, a peer in our system maintains dynamic data.

8 Experimental Study

In this section, we built a peer-to-peer simulator to evaluate the performance of our proposed system over large-scale networks. The simulator simulates an unstructured P2P network by randomly producing input parameters including (1) the number of peers, (2) k value, (3) various distribution, (4) the number of groups of peers, and (5) the number of update peers. We used the number of conveying message (labeled as "# of message") to measure the performance of the system and each message consists of a 2D location and a value.

To evaluate the cost of query processing, we tested the network with different number of peers N from 200 to 3000. Each peer contains a set of data from 1K to 200K with two dimensional location values and one non-location value. We generated two data sets. Data set 1 conforms to uniform distribution and data set 2 conforms to normal distribution. We constructed all kinds of the physical relationships among the peers, which are the semi-surrounded, embedded, and non-overlapping relationship. We randomly built the logical relationship among the peers. All the approaches were implemented in C++ and run on Intel XEON(TM) 3.2GHz dual-CPU with 2GB RAM on Windows 2003 Server. For each setting, we tested algorithms by running it 10 times to compute the average result.

Comparison of Different k-NN Searches. We compared the communication cost (represented as # of messages) of getting *k-NN* results using different search strategies. We ran 10 different *k-NN* queries whose locations and predicate values are randomly specified. Besides flooding search, we tested two radius-convergence approaches and two radius-expansion approaches. Radius-convergence approaches include random search and RA-based search[35], whereas radius-expansion approaches include exhausted search every dominate groups (EX in short) and partition-based search (PA in short). We used two data sets with the uniform and normal distributions, respectively. In addition, we also compared three straightforward approaches according to the ranking of the minimal and the maximal geographical distances and the shortest path in logical graph, respectively.

Fig.10 shows the conveying of messages of these algorithms uses two data sets conforming to the uniform and normal distributions. We let the number of peers vary between 500 and 3000, and ran 10 different 100-NN queries. Figs.10(a) and 10(b) compare five approaches using data set 1. They show that flooding search costs

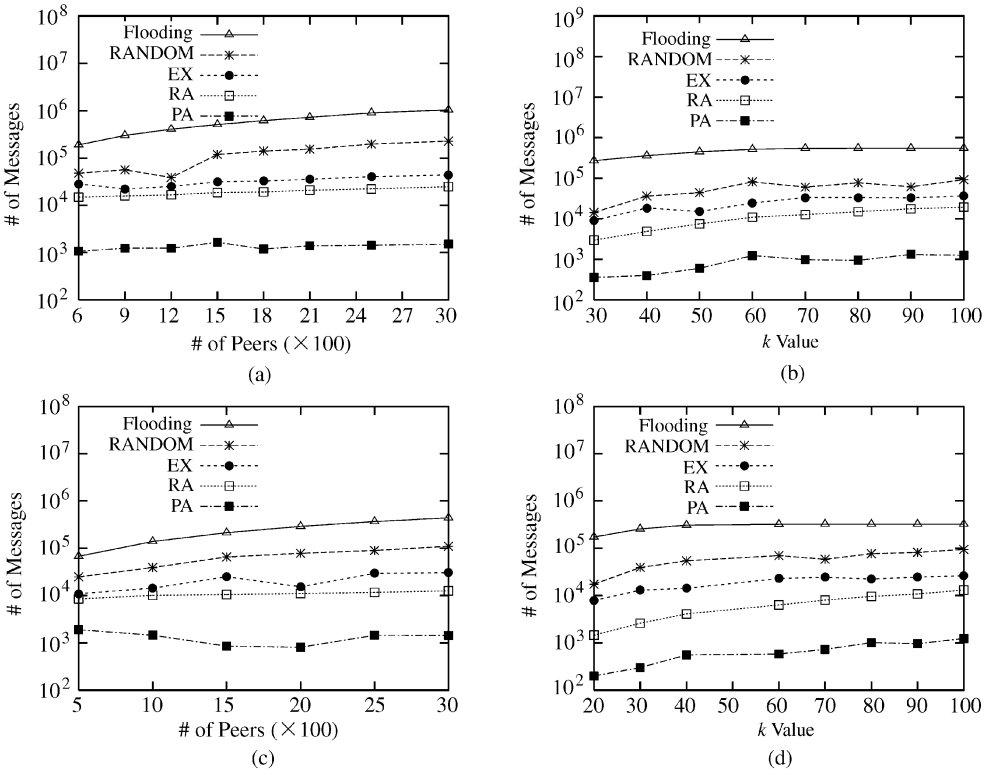


Fig.10. Comparison of different *k-NN* searches. (a) (b) Uniform distribution. (c) (d) Normal distribution.

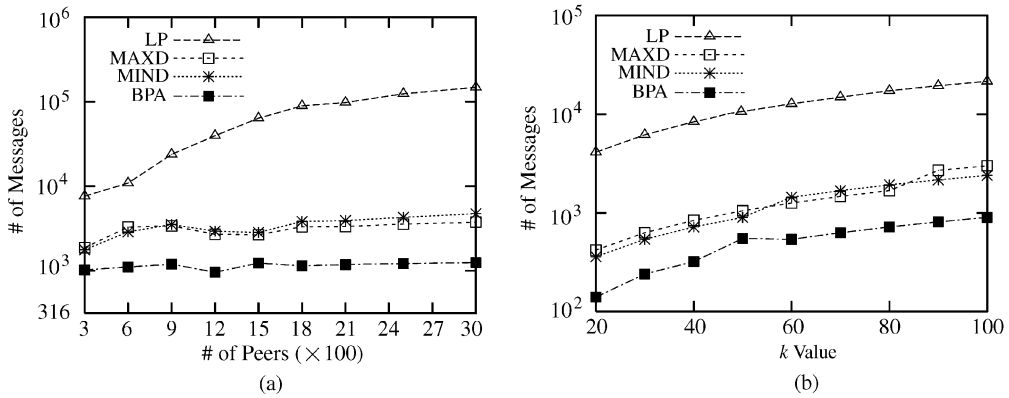


Fig.11. Effectiveness of geographical distances and logical paths.

the most conveying messages, whereas PA uses the least conveying messages. Similar to RA approach, our proposed PA approach is independent on the number of peers. Fig.10(c) shows the similar result using data set 2. Figs.10(b) and 10(d) show the number of conveying messages when fixing the number of peers to 1000 and varying k from 20 to 100. They all show that PA outperforms the other approaches, and PA always uses least messages to get *k-NN* nearest results.

We further compared PA with three heuristic approaches, which are (i) peers with shortest logical paths take precedence (LP in short), (ii) peers with shortest minimum distances to q take precedence (MIND in short), and (iii) peers with shortest maximum distances to q take precedence (MAXD in short). Fig.11 shows the number of conveying messages on data set 1. Fig.11(a) shows the result of 100-NN queries when the number of peers (i.e., the size of the system) vary from 300 to 3000. Fig.11(b) shows the result of *k-NN* queries when varying k from 20 to 100 over 1000 peers. LP uses the most messages to get the results since a shorter logical path does not mean a shorter geographical distance to q. MIND and MAXD have the similar performances since both of them give priority to geographical dis-

tances. When we increased the number of peers in the system, MIND uses less messages than MAXD since MIND takes larger probabilities to get closer data than MAXD. The reason why PA always uses the least messages is that PA considers not only geographical distances but also logical probing paths.

Comparison of Filtering Capabilities. We tested the capabilities of filtering peers using PA and RA approaches on data set 1. We first changed the number of k values from 20 to 70. Fig.12(a) shows the filtering capability decreased when k values is increased. RA has less filtering capability than PA. The reason is RA is a radius-convergence approach that gradually shrink query radius. It keeps probing at least k peers for each iteration until all peers in the shrinking query radius have been probed. Whereas, PA is a radius-expansion approach that when k closest data is met, it can stop. Fig.12(b) shows the filter capabilities when varying the number of peer groups. It shows that PA has better filtering capability than RA. When the peer group number increased to 16, the filtering capability of PA climbs to a peak value 52%, whereas RA has no peak value. It proved that the filtering capability of PA depends on the chosen number of groups in all dominate group set, whereas RA does not since RA only considers the first k ranked peers using geographical distances and logical graphs.

Effectiveness of Histogram. Now we show the effectiveness of histogram. We compared two PA algorithms, one is the approach by which probe and query peers are executed at the same round (i.e., without histogram), the other is the approach by which histograms are used

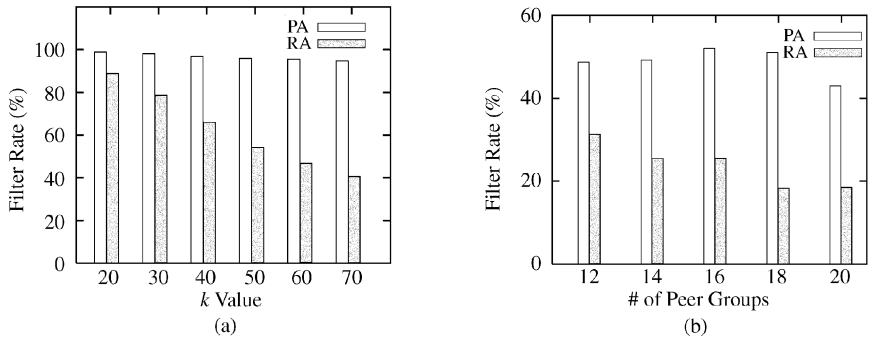


Fig.12. Filtering capabilities.

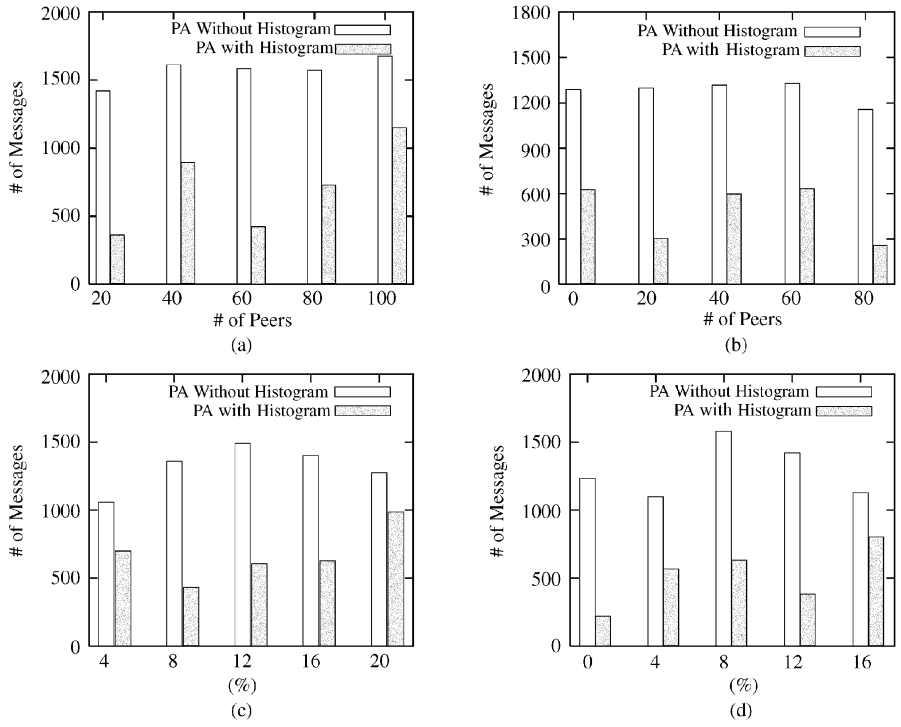


Fig.13. Effectiveness of histogram for continuous query. (a) Insertion. (b) Deletion. (c) Expansion. (d) Shrinking.

during the first probe round. On the basis of the collected histograms from peers, we calculated an estimated query radius to process the query. Fig.13 shows that for different numbers of peers, the histogram technique always helps to save significant conveying messages than the PA approach without histogram.

DM vs. R-Tree Experiment. We compared the update performance of peers' join and leave using our proposed domination model (DM) and the representative R-tree, respectively. We applied the insertion and deletion operations on DM and the R-tree. Fig.14 shows the CPU time as a function of the number of update peers using data set 1. We used 2500 as the number of peers that were managed by DM and R-tree, respectively. We randomly selected a certain number of update peers (insertion or deletion), let it vary from 300 to 1800. Fig.14 shows that DM spends much less CPU time on maintaining updates than R-tree when peers dynamically change.

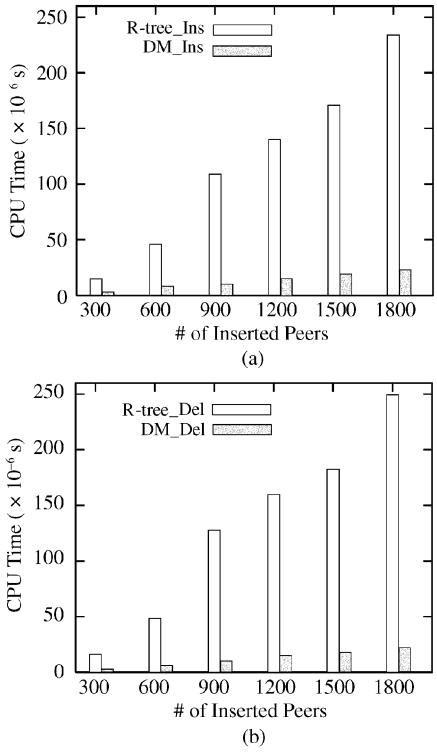


Fig.14. Comparison of R-tree and DM model.

To summarize, the PA with a histogram outperforms other approaches significantly under all settings. Furthermore, according to our experiments, we obtained the final results: (i) the proposed PA approach outperforms most of the other heuristic algorithms, (ii) the novel adaptive histogram can save more communication cost, and (iii) our technique can efficiently process continuous *k-NN* queries in a distributed, dynamic, and large scale environment.

9 Conclusion

This paper has investigated the new problem of processing the continuous constrained *k-NN* queries over an unstructured P2P system, and proposed a domination model to dynamically partition the search space and several methods to efficiently filter peers in the search space. Moreover, this paper has also extended our techniques to cope with updates of peers and their data. The experimental results on the two synthetic data sets have shown that the PA with a histogram outperforms the other approaches significantly under all the cases. Furthermore, the experiments have also shown that (i) the algorithms proposed outperform most of the other heuristic algorithms, (ii) the novel adaptive histogram can save more communication cost, and (iii) our technique can efficiently process continuous *k-NN* queries in a distributed, dynamic and large scale environment. Future work includes incorporating local index to the proposed domination model to efficiently maintain updates in the unstructured P2P system.

References

[1] Hjaltason G R, Samet H. Distance browsing in spatial databases. *ACM Trans. Database Syst.*, 1999, 24(2): 265–318.

[2] Korn F, Sidiropoulos N, Faloutsos C *et al.* Fast nearest neighbor search in medical image databases. In *Proc. 22nd Int. Conf. Very Large Data Bases*, Mumbai (Bombay), India, 1996, pp.215–226.

[3] Mouratidis K, Yiu M L, Papadias D *et al.* Continuous nearest neighbor monitoring in road networks. In *Proc. 32nd Int. Conf. Very Large Data Bases*, Seoul, Korea, 2006, pp.43–54.

[4] Seidl T, Kriegel H. Optimal multi-step *k*-nearest neighbor search. In *Proc. ACM SIGMOD Int. Conf. Management of Data*, Seattle, Washington, USA, 1998, pp.154–165.

[5] Wang Y F, Hori Y, Sakurai K. Characterizing economic and social properties of trust and reputation systems in P2P environment. *Journal of Computer Science and Technology*, 2008, 23(1): 129–140.

[6] V T de Almeida, R H Güting. Using dijkstra's algorithm to incrementally find the *k*-nearest neighbors in spatial network databases. In *Proc. the 2006 ACM Symposium on Applied Computing* (*SAC*), Dijon, France, 2006, pp.58–62.

[7] Hu H, Lee D L, Xu J. Fast nearest neighbor search on road networks. In *Proc. 10th International Conference on Extending Database Technology* (*EDBT*), Munich, Germany, *Lecture Notes in Computer Science*, 3896, 2006, pp.186–203.

[8] Kolahdouzan M R, Shahabi C. Voronoi-based *k* nearest neighbor search for spatial network databases. In *Proc. 30th Int. Conf. Very Large Data Bases*, Toronto, Canada, 2004, pp.840–851.

[9] Yiu M L, Papadias D, Mamoulis N, Tao Y. Reverse nearest neighbors in large graphs. *IEEE Trans. Knowl. Data Eng.*, 2006, 18(4): 540–553.

[10] Jagadish H V, Ooi B C, Vu Q *et al.* Vbi-tree: A peer-to-peer framework for supporting multi-dimensional indexing

schemes. In *Proc. 22th Int. Conf. Data Engineering*, Atlanta, GA, USA, 2006, p.34.
[11] Jagadish H V, Ooi B C, Vu Q H. Baton: A balanced tree structure for peer-to-peer networks. In *Proc. 23rd Int. Conf. Very Large Data Bases*, Trondheim, Norway, 2005, pp.661–672.
[12] Tanenbaum A S. Computer Networks. Third edition, Prentice Hall PTR, ISBN 0-13-349945-6, 1996, pp.355–359.
[13] Silberstein A, Braynard R, Ellis C *et al.* A sampling-based approach to optimizing top-k queries in sensor networks. In *Proc. 22nd Int. Conf. Data Engineering*, Atlanta, GA, USA, 2006, p.68.
[14] Robinsy G, Zelikovskyz A. Improved Steiner tree approximation in graphs. In *Proc. the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms* (*SODA*), San Francisco, CA, USA, 2000, pp.770–779.
[15] Saltenis S, Jensen C S, Leutenegger S T *et al.* Indexing the positions of continuously moving objects. In *Proc. ACM SIGMOD Int. Conf. Management of Data*, Dallas, Texas, USA, 2000, pp.331–342.
[16] Tao Y, Papadias D. MV3R-tree: A spatio-temporal access method for timestamp and interval queries. In *Proc. 27th Int. Conf. Very Large Data Bases*, Rome, Italy, 2001, pp.431–440.
[17] Ratnasamy S, Francis P, Handley M *et al.* A scalable content addressable network. In *Proc. the 2001 ACM SIGCOMM Conference*, Santa Barbara, CA, USA, 2001, pp.161–172.
[18] Stoica I, Morris R, Karger D *et al.* Chord: A scalable peer-to-peer lookup service for Internet applications. In *Proc. the 2001 ACM SIGCOMM Conference*, Santa Barbara, CA, USA, 2001, pp.149–160.
[19] Rowstron A, Druschel P. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Proc. the 18th IFIP/ACM International Conf. Distributed Systems Platforms*, Heidelberg, Germany, *Lecture Notes in Computer Science*, 2218, 2001, pp.329–350.
[20] Zhao B Y, Kubiatowicz K, Joseph A D. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. *Technical Report*, University of California, Berkeley, 2001.
[21] Aberer K, Mauroux P C, Datta A, Despotovic Z, Hauswirth M, Punceva M, Schmidt R. P-grid: A self-organizing structured p2p system. *SIGMOD Record*, 2003, 32(3): 29–33.
[22] Crainiceanu A, Linga P, Machanavajjhala A *et al.* P-ring: An index structure for peer-to-peer systems. *Technical Report*, TR2004-1946, Cornell University, 2004.
[23] Bharambe A, Agrawal M, Seshan S. Mercury: Supporting scalable multi-attribute range queries. In *Proc. the 2004 ACM SIGCOMM Conference*, Portland, Oregon, USA, 2004, pp.329–350.
[24] Gnutella. http://www.gnutella.com/.
[25] Guttman A. R-trees: A dynamic index structure for spatial searching. In *Proc. ACM SIGMOD Int. Conf. Management of Data*, Boston, Massachusetts, USA, 1984, pp.47–57.
[26] Bentley J. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 1975, 18(9): 509–517.
[27] Lin K I, Jagadish H V, Faloutsos C. The tv-tree: An index structure for high-dimensional data. *The VLDB Journal*, 1994, 3(4): 517–542.
[28] Katayama N, Satoh S. The sr-tree: An index structure for high-dimensional nearest neighbor queries. In *Proc. ACM SIGMOD Int. Conf. Management of Data*, Tucson, Arizona, USA, 1997, pp.369–380.
[29] Berchtold S, Keim D A, Kriegel H P. The x-tree: An index structure for high-dimensional data. In *Proc. 22nd Int. Conf. Very Large Data Bases*, Bombay, India, 1996, pp.28–39.
[30] Uhlmann J K. Satisfying general proximity/similarity queries with metric trees. *Information Processing Letter*, 1991, 40(4): 175–179.
[31] Brin S. Near neighbor search in large metric spaces. In *Proc. 21st Int. Conf. Very Large Data Bases*, Zurich, Switzerland, 1995, pp.574–584.
[32] Navarro G. Searching in metric spaces by spatial approximation. *The VLDB Journal*, 2002, 11: 28–46.
[33] Ciaccia P, Patella M, Zezula P. M-tree: An efficient access method for similarity search in metric spaces. In *Proc. 23rd Int. Conf. Very Large Data Bases*, Athens, Greece, 1997, pp.426–435.
[34] Bozkaya T, Ozsoyoglu M. Indexing large metric spaces for similarity search queries. *ACM Trans. Database Syst.*, 1999, 24(3): 361–404.
[35] Fagin R, Lotem A, Naor M. Optimal aggregation algorithms for middleware. In *Proc. ACM SIGACT-SIGMOD Symp. Principles of Database Systems*, Santa Barbara, California, USA, 2001, pp.102–113.

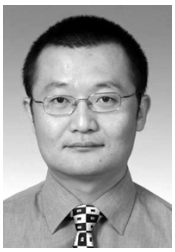
Bin Wang received the Master's degree in computer science from the Northeastern University in 2001. He is currently pursuing the Ph.D. degree in computer science at the Northeastern University under the supervision of Professor Guoren Wang. His research interests include design and analysis of algorithms, databases, data quality, and distributed systems. He is a member of CCF.

Xiao-Chun Yang received the Ph.D. degree in computer science from Northeastern University, China, in 2001. She is an associate professor in the Department of Computer Science at Northeastern University, China. Her research interests are in the areas of data quality, data privacy, and distributed data management for sensor networks and P2P networks. She has received a China Program Award for New Century Excellent Talents in Universities, a China National Natural Science Foundation Grant, a Fok Ying Tong Education Foundation Grant, and a China Ministry of Education Grant. She is a member of the ACM, the IEEE Computer Society, and a senior member of CCF.

Guo-Ren Wang received the B.Sc., M.Sc. and Ph.D. degrees from Northeastern University, China, in 1988, 1991 and 1996, respectively. He is currently the leader of Modern Information Technology Team, the director of Institute of Computer System, the vice dean of College of Information Science and Engineering, Northeastern University, and a senior member of CCF. His research interests are XML data management, query processing and optimization, bioinformatics,

high-dimensional indexing, parallel database systems, and P2P data management. He has published more than 100 research papers.

Ge Yu received his B.E. degree and M.E. degree in computer science from Northeastern University of China in 1982 and 1986, respectively, Ph.D. degree in computer science from Kyushu University of Japan in 1996. He has been a professor at Northeastern University of China since 1996. He is a member of IEEE, ACM, and a senior member of CCF. His research interests includes database theory and technology, distributed and parallel systems, embedded software, network information security.



Lei Chen received his B.S. degree in computer science and engineering from Tianjin University, China, in 1994, the M.A. degree from Asian Institute of Technology, Thailand, in 1997, and the Ph.D. degree in computer science from University of Waterloo, Canada, in 2005. He is now an assistant professor in the Department of Computer Science and Engineering at Hong Kong University of Science and Technology. His research interests include multimedia and time series databases, probabilistic databases, graph databases, sensor and peer-to-peer databases.

X. Sean Wang received his Ph.D. degree in computer science from the University of Southern California, Los Angeles, California, USA, in 1992. He is currently the Dorothean chair professor in computer science at the University of Vermont, Burlington, Vermont, USA. He has published widely in the general areas of databases and information security, and was a recipient of the US National Science Foundation Research Initiation and CAREER awards. His research interests include database systems, information security, data mining, and sensor data processing.

Xue-Min Lin is a professor in the School of Computer Science and Engineering, the University of New South Wales. Currently, he is the head of database research group. Dr. Lin received his Ph.D. degree in computer science from the University of Queensland, Australia, in 1992 and his B.Sc. degree in applied math from Fudan University, China, in1984. During 1984∼1988, he studied for Ph.D. degree in applied math at Fudan University. Dr. Lin's principle research areas cover databases and graph visualization.