Reverse Nearest Neighbor Search in Peer-to-Peer Systems

Dehua Chen, Jingjing Zhou, and Jiajin Le

Col. of Computer Science, University of Donghua
P. O. Box 324, 200051, Shanghai, PRC
lydehua@mail.dhu.edu.cn
lejiajin@dhu.edu.cn

Abstract. Given a query point *Q*, a *Reverse Nearest Neighbor* (**RNN**) Query returns all points in the database having *Q* as their nearest neighbor. The problem of **RNN** query has received much attention in a centralized database. However, not so much work has been done on this topic in the context of Peer-to-Peer (P2P) systems. In this paper, we shall do pioneering work on supporting distributed **RNN** query in large distributed and dynamic P2P networks. Our proposed **RNN** query algorithms are based on a distributed multi-dimensional index structure, called *P2PRdNN*-tree, which is relying on a super-peer-based P2P overlay. The results of our performance evaluation with real spatial data sets show that our proposed algorithms are indeed practically feasible for answering distributed **RNN** query in P2P systems.

1 Introduction

The problem of *Reverse Nearest Neighbor* (RNN) Query [1,2,3,4,5,6,7,8,9,10] is to retrieve all data points in given multi-dimensional data sets whose Nearest Neighbor (NN) is a given query point. Although RNN is a complement of NN problem it is more complex than NN problem. The solutions from NN query cannot be directly applied to RNN query. This is because of the asymmetric relationship between NN/RNN: if a data point p is an RNN(q) (q is the nearest neighbor of p), it does not imply that p is the nearest neighbor NN(q) of q. The RNN problem has recently received considerable attention in the context of centralized database system due to its importance in a wide range of applications such as decision support system, profile-based marketing, document databases etc.

Nowadays, Peer-to-Peer (P2P) systems have become popular for sharing resources, information and services across a large number of autonomous peers in Internet. Especially, the applications of sharing multi-dimensional data (e.g. spatial data, documents, image files) in P2P systems are now being widely studied in the literatures [11,12,13,14,15,16,17]. However, most of these applications focus mainly on two types of queries: Range query and Nearest Neighbor (NN) query on the distributed data sets. And not so much effort is taken to support RNN search in such large distributed and ad-hoc environment. However, we believe that like its importance in the centralized database system, RNN query will become a practical

H. Legind Larsen et al. (Eds.): FQAS 2006, LNAI 4027, pp. 87 – 96, 2006.

and important class of queries in P2P systems. Let us first consider an example in the P2P Geographic Information System (GIS) application. Suppose a large-scale chain supermarket is to open up a new supermarket at a location, the RNN query can be used to find the subset of existing supermarkets will be affected by the new supermarket, assuming people choose the nearest supermarket to consume. Another example is that when a new document is inserted into a P2P digital library, the RNN query can be used to ask the subset of authors of other documents who will find the new document interesting based on similarity to their documents. Therefore, this paper will investigate RNN search in distributed and dynamic P2P systems.

Like most of previous researches for RNN query in centralized database system, our proposed methods also build on tree-based multi-dimensional index structures (e.g. the R-tree family [18,19,20]). However, instead of maintaining a centralized multi-dimensional index in one centralized server, we propose a distributed multi-dimensional index, called P2PRdNN-tree, supported by a super-peer-based P2P overlay network. The P2PRdNN-tree structure enables efficient RNN search in large distributed environment. Like Rdnn-tree [2] proposed for centralized database context, our proposed distributed P2PRdNN-tree index structure stores extra information about nearest neighbor of data points in tree nodes. The extra information can efficiently reduce the search space and network communication

The remainder of this paper is organized as follows: Section 2 overviews the previous work. Section 3 presents our proposed super-peer-based P2P overlay and P2PRdNN-tree structure. Section 4 presents our proposed distributed RNN search algorithms. Section 5 provides experimental results and Section 6 summarizes our work.

2 Related Work

In Section 2.1, we shall briefly describe previous work on RNN query in centralized database systems. Section 2.2 overviews multi-dimensional data sharing in P2P systems.

2.1 RNN Search in Centralized Database Systems

Algorithms for processing RNN query in centralized databases can be classified into two categories depending on whether they require pre-computation or not.

The problem of RNN was first studied in [1]. The idea of the authors is to pre-compute, for each data point d, the distance dnn to its nearest neighbor NN(d). Thus, each data point is represented as a circle, whose center is the data point and whose radius is its dnn. Besides the R-tree that indexes the original data point, a separate R-tree is maintained which indexes the sets of such circles. The problem of finding RNN of a query point Q is then reduced to finding the circles that contain Q.

In order to avoid maintaining two separate R-trees, [2] combines the two indexes in the Rdnn-tree (R-tree containing Distance of Nearest Neighbors) index structure. Rdnn-tree differs from standard R-tree by storing extra information about NN of the data points for each tree node: for every leaf node, its record stores dnn, and for every non-leaf node, it record stores max_dnn (the maximum distance from every point in

the sub-tree to its nearest neighbor). Therefore it requires prior computation of NN for each data point. The Rdnn-tree benefits RNN queries as follows. Let a non-leaf node be *N*, and let the query point be *Q*. If the distance between *Q* and the *MBR* (Minimal Bounding Rectangle) of *N* is bigger than *N*'s *max_dnn*, there is no need to search the sub-tree rooted by *N*. Inspired by the idea of Rdnn-tree, our proposed distributed multi-dimensional index structure also maintains extra information about the nearest neighbor in each tree node for assisting in efficient distributed RNN search.

There are several methods without relying on pre-computation. The approach of [3] divides the (2D) data space around the query point *Q* into six 60° regions, such that the only candidate of the RNN of *Q* in each region is exactly the NN of *Q*. So [3] finds the six NNs, and then check to see if each of them really considers *Q* as NN. [8] introduces another approach. Its idea is to find the NN (say o_1) of a query point *Q* first. Then consider the bisector of *Q* and o_1. All data points on the side of o_1 (except o_1 itself) can be pruned, since their distances to o_1 is no more than the distances to *Q*. Next, in the unpruned space, the NN to *Q* is found, and the space is further pruned. Finally, the unpruned space does not contain any data point. The only candidates of RNN are the identified NNs. The refinement step, which removes false positives, uses the previously pruned MBRs so that no tree node is visited twice throughout the algorithm.

Above we have reviewed the traditional monochromatic RNN query. There are other versions of RNN query have been proposed. [9][4] propose the solutions for *bichromatic* RNN queries where, given a set *Q* of queries, the goal is to find the points $d \in D$ that are closer to some $q \in Q$ than any other point of *Q*; [5] investigates *continuous* RNN queries on spatiotemporal data; [6] examines *stream* RNN queries where data arrive in the form of stream, and the goal is to report aggregate results over the RNNs of a set of query points.

2.2 Multi-dimensional Data Sharing in P2P Systems

The sharing of multi-dimensional data in P2P systems has become popular recently. CAN [21] can be regarded as the first P2P system supporting the sharing of multi-dimensional data since it has the same structure as the kd-tree[22] and grid-file[23]. pSearch [14], a P2P system based on CAN, is proposed for retrieving documents that are modeled as points in multi-dimensional space. [27] has proposed another system also based on CAN for supporting range query by including the ranges into hash functions. Most other systems such as [15] use space filling curves to map multi-dimensional data to one dimensional data. SkipIndex [16] is based on skip graph [28], which aims to support high dimensional similarity query. More recently, several distributed multi-dimensional index structures have been proposed in the literatures. [13] proposed an R-tree-based indexing structure for P2P systems in the context of sensor network. The proposed index structure in [13] is designed for optimize NN search. P2PR-tree [12] is another distributed multi-dimensional index structure based on R-tree. P2PR-tree is well designed for optimizing window query. [11] proposed VBI-tree, a new Peer-to-Peer framework based on a balanced tree structure overlay, which can support extensible centralized mapping methods and query processing based on a variety of multidimensional tree structures, including R-Tree, X-Tree [24], SSTree [25], and M-Tree [26].

3 P2PRdNN-Tree Structure

In this section, we first present a super-peer-based P2P overlay network and then propose P2PRdNN-tree implemented on top of such P2P overlay.

3.1 Super-Peer-Based Overlay

Peers in a P2P system may be organized by various network overlays. Considering the fact that peers in the network often vary widely in bandwidth and computing capacity, we organize peers into a super-peer-based P2P topology (shown as in Fig. 3.1).

In such super-peer-based P2P network infrastructure, a small subset of peers with relatively high stability and relatively high computing capacity are designated as super-peers. As we can see in the following sections, super-peers take over a lot of important responsibilities such as routing query and answer messages, initiating and maintaining local P2PRdNN-tree structure, distributing and executing of query plan. For simplicity, we connect super-peers with a main channel that acts as a broadcast routing mechanism and can be implemented in many different ways. Certainly, super-peers can also be arranged in more complex topology such as Hypercup [29].

Each peer storing and maintaining a set of multi-dimensional data points connects directly to one super-peer in the network. Peers can join and leave the system in any time and have relatively lower computing power.

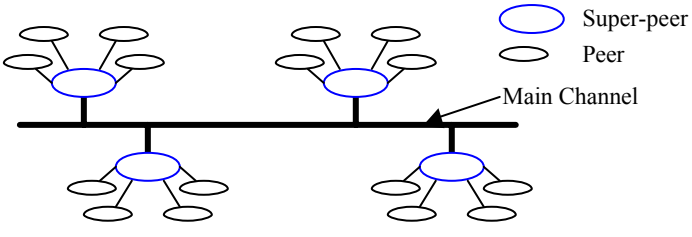

Fig. 3.1. Super-peer-based P2P overlay

3.2 P2PRdNN-Tree

In this section, we shall present the structure of P2PRdNN-tree based on our proposed super-peer-based P2P overlay in the above section.

Assume that each peer, say p, stores and manages a set D_p of n dimensional data points concerning a certain region of the n dimensional space. The region can be expressed in the form of the MBR_p that bounds the data points in D_p. We can also say that the peer p is responsible for the MBR_p. For each data point, say d, in the data set D_p, we compute the distance $L\_dnn_{Dp}(d)$ to its local nearest neighbor NN(d) in D_p. Please note that the local nearest neighbor of a point, defined here, may not be the real (global) nearest neighbor in all data sets available on the network. The real (global) nearest neighbor of a point may locate in other peer. This can be illustrated in Fig. 3.2. The real nearest neighbor b of data point d locates in peer B, and point a is the local nearest

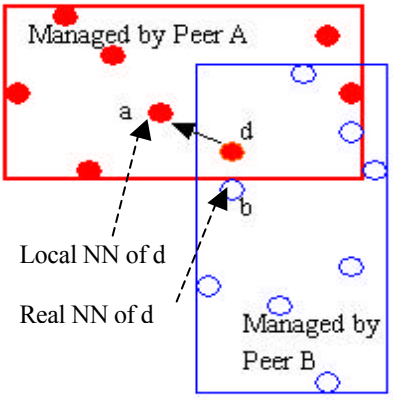

Fig. 3.2. The concept of Local nearest neighbor

neighbor of *d* in peer *A*. However, in order to avoid introducing extra network traffic, we do not compute here the distance between the point and its real nearest neighbor.

For each peer in the network, it should send its information to its corresponding super-peer. The information include the location of the peer (e.g. IP address and port), its responsible MBR and *max_dnn*=max{L_dnn_{Dp}(d)}. The super-peer shall initiate a P2PRdNN-tree structure based on the information from the peers connecting to it. We present the structure of P2PRdNN-tree in the following paragraphs.

In case of P2PRdNN-tree structure, each peer is assigned one leaf node of P2PRdNN-tree. Like the Rdnn-tree, the leaf node of P2PRdNN-tree contains entries of the form (*ptid*, L_*dnn*), where *ptid* refers to an *n* dimensional point and L_*dnn* is the distance from the point to its local nearest neighbor. The entries of a leaf node of tree are stored in the responsible peer.

As far as the super-peer is concerned, it does not hold the leaf level of P2PRdNN-tree. It maintains the non-leaf nodes of P2PRdNN-tree. Similar to Rdnn-tree, each non-leaf node of P2PRdNN-tree stores a set of entries of the form (*ptr*, *rect*, *max_dnn*). *ptr* is the address of a child node in the tree. If *ptr* points to a leaf node, the address refers to the location of the responsible peer and *rect* is the MBR of the leaf node. If *ptr* points to a non-leaf node, the *rect* is the MBR of all rectangles that are entries in the child node. *Max_dnn* is the maximum distance from every data point in the subtree to its nearest neighbor. For the root node of P2PRdNN-tree, it stores an additional entry of the form of (G_MBR, G_maxdnn), where G_MBR refers to the general MBR bounding all points managed by its connected peers, and G_maxdnn is the maximum max_dnn of all points in G_MBR.

Thus, for each super-peer and its directly-connecting peers, they maintain together a local P2PRdNN-tree indexing the data sets stored in the peers. Fig.3.3 shows a local P2PRdNN-tree structure. In the Fig.3.3 (b), Peer A ~H manage the data points of the regions a ~h respectively. In the local P2PRdNN-tree from Fig.3.3 (a), leaf nodes a ~h are managed by Peer A ~H respectively. The super-peer is responsible for maintaining other part of the tree (the non-leaf nodes i ~o).

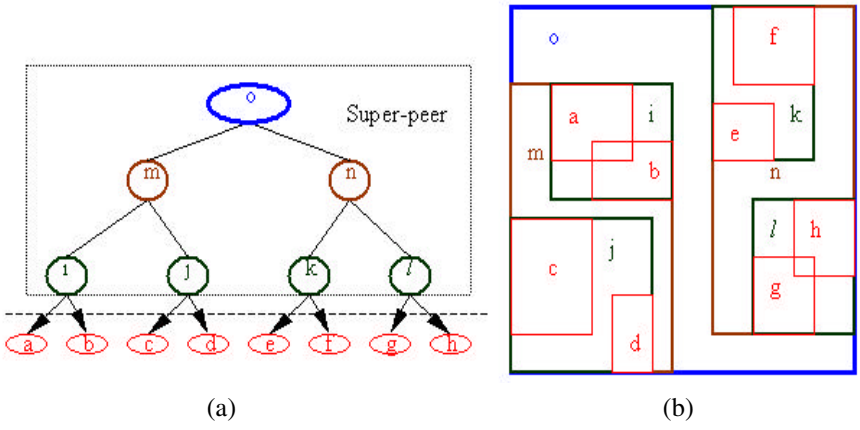

Fig. 3.3. The P2PRdNN-tree index structure

4 RNN Search with P2PRdNN-Tree

In this section, we shall discuss how to process distributed RNN search based on the P2PRdNN-tree index structure presented in the above section.

Our distributed RNN search algorithm operates in two steps: (1) the filter step retrieves a candidate RNN set (discussed in section 4.1), and (2) the verification step eliminates false hits and reports the actual RNNs (discussed in section 4.2). The introduction of the second step has something to do with the definition of L_*dnn* presented in above section.

According to the distributed nature of the query and the P2P network, each step in the algorithm consists of three parts that are respectively executed by

(i) The super-peer initially receiving the query,
(ii) Other partitioning super-peers in the network,
(iii) And the local peers at each partitioning super-peer.

Let us now present the algorithm to answer a RNN query Q posed by peer P to super-peer *SP*, also called the initiating super-peer. The algorithm is entirely controlled by super-peer *SP*, which, whenever necessary, poses the requests to its connected peers and other super-peer in the network during the execution.

We shall study two steps in our proposed algorithm respectively as follows.

4. 1 Filter Step

When super-peer *SP* receives a RNN query Q from one of its connected peers. It first broadcasts via the main channel a query message in a form of (Q, Query_id) where Q indicates the query is a RNN query and Query_id is the unique identifier of the query. The Query_id prevents one super-peer or peer from receiving a query twice. Every super-peer (including *SP*) receiving the query message computes the distance Dist(Q, G_MBR) between Q and its G_MBR from the root node of its local P2PRdNN-tree. If Dist(Q, G_MBR) is greater than its corresponding G_maxdnn, we can conclude that

any point in the G_MBR is not closer to Q than its nearest neighbor. In other words, there is no reverse nearest neighbor of Q in the G_MBR. Thus, we need not to search the peers connecting to this super-peer for reverse nearest neighbor. Then the super-peer discards the query message. Otherwise, the super-peer accepts the query and performs the filter algorithm (shown in Fig.4.1) by branch-and-bound traversing the local P2PRdNN-tree. For those super-peers accepting the query message, we call them query-partitioning super-peer (partitioning super-peer for short). Please note that the initiating super-peer may be partitioning super-peer.

Determining_candidates_of _RNN (Node *n*, Query *Q*)

If *n* is leaf node of P2PRdNN-tree, the partitioning super-peer sends a request to the peer who is responsible for the leaf node. And at the responsible peer, for each entries (*ptid*, L_*dnn*),
if Dist(*Q*, *ptid*)<L_*dnn*, outputs the data point referred by *ptid* as one of the candidate RNNs of *Q*, and sends the candidate to the initiating super-peer *SP*.

If *n* is non-leaf node of P2PRdNN-tree, then for each branch B=(*ptr*, *rect*, *max_dnn*),
if Dist(*Q*,*rect*)<*max_dnn*, call Determine_candidate_of_RNN (B.*ptr*,*Q*)

Fig. 4.1. The filter algorithm

4.2 Verification Step

When each candidate retrieved during the filter step arrives at the initiating super-peer *SP*, we need to verify whether the candidate is actual RNN of Q. For each candidate c, the initiating super-peer *SP* broadcasts via main channel a Range Query in the form of [$Q(c,r)$, Query_id] where $Q(c,r)$ refers to the query is a kind of circle range query with center point c and radiu r=Dist(c,Q), and Query_id is the unique identifier of the query.

Every super-peer (including *SP*) receiving the range query message checks whether its G_MBR covers or intersects the region of $Q(c,r)$. If the result is false, then the super-peer discards the range query message. Otherwise, the super-peer accepts the range query and performs the verification algorithm by traversing the local P2PRdNN-tree using any existing range query algorithm on R-tree. Similarly, we call those super-peers accepting the query message as partitioning super-peer. The verification step terminates if all the partitioning super-peers report no points in the query region or if there is a partitioning super-peer reports points in the query region. If the former condition is satisfied, then the candidate c is an actual RNN of Q and is reported to the query peer by the initiating super-peer *SP*. Otherwise, the candidate c is discarded by *SP*.

5 Experiments

We conducted simulation experiments to evaluate the performance of our proposed algorithms. Our simulation environment comprised a 100-node computer cluster. In

order to compare our work meaningfully against the centralized database context, we implement two different network topologies for organizing these machines: the first is Super-Peer (SP) based topology for P2P environment; the second is star topology for Client/Server (C/S) architecture. Our performance study was conducted using a real-world dataset known as *Tiger* data files.

For super-peer based topology, all simple peers have exactly one connection to a super-peer. The super-peers form their own P2P network. For the experiments described in this paper we use main channel to connect all super-peers. In our simulation experiments, we let each simple peer manage multiple leaf nodes of local P2PRdNN-tree. Each leaf node has more than 100000 spatial objects.

For star topology, all spatial objects are stored in one server to which all other peers connect. In the server, we index all objects by Rdnn-tree [2].

The simulation experiment results are shown in Fig.5.1. In the figure, the x-axis represents an inter-arrival rate of *n* RNN queries which implies *n* RNN queries were issued in the entire system every second. The y-axis represents the average completion time that indicates the average time taken for each query to return all answer from relevant peers. From the results in Fig.5.1, we find that as the inter-arrival rate of queries increases, the performance of C/S drops more quickly than SP. This occurs because in C/S *every* query has to be routed only to the centralized server. As a result, there are large job queues at the centralized server, thereby causing significantly increased waiting times at the server that ultimately causes severely the completion times to increase. In contrast, the decentralized nature of SP implies that query execution is performed in a distributed fashion, thereby ensuring the absence of any serious execution bottlenecks.

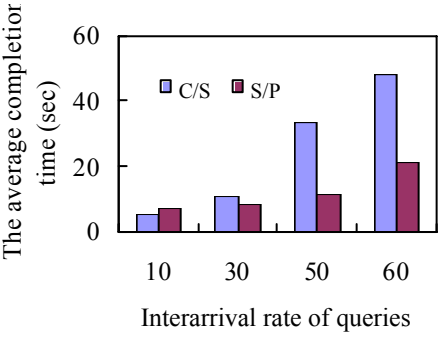

Fig. 5.1. The performance comparison between C/S and S/P

6 Conclusions and Future Work

In this paper, we have made the pioneering investigation on distributed Reverse Nearest Neighbor search in P2P systems. In our work, our proposed RNN search algorithms are based on the P2PRdNN-tree, a P2P version of Rdnn-tree, which is well-suited for RNN search in P2P environment. Meanwhile, our proposed P2PRdNN-tree also supports Range Query and Nearest Neighbor Query since it is also a variation of R-tree.

This paper does not discuss the issue of the update of our distributed P2PRdNN-tree when the evolutions of network and data take place. However, this issue is interesting and challenging. We plan to Study this issue in our future work.

This paper focuses on the traditional monochromatic RNN query in P2P systems. We also plan to investigate other RNN queries such as *bichromatic* RNN in distributed environment.

Our proposed solutions work well for low dimensional data. We wish to explore how to adjust our solutions for performing RNN search on high-dimensional data in P2P systems.

References

1. F. Korn and S. Muthukrishnan: Influence Sets Based on Reverse Nearest Neighbor Queries. In SIGMOD, 2000.
2. C. Yang and K. I. Lin: An Index Structure for Efficient Reverse Nearest Neighbor Queries. In ICDE, 2001.
3. I. Stanoi, D. Agrawal, and A. E. Abbadi: Reverse Nearest Neighbor Queries for Dynamic Databases. In SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, 2000.
4. I. Stanoi, M. Riedewald, D. Agrawal, and A. E. Abbadi: Discovery of Influence Sets in Frequently Updated Databases. In VLDB, 2001.
5. R. Benetis, C. S. Jensen, G. Karciauskas, and S. Saltenis: Nearest Neighbor and Reverse Nearest Neighbor Queries for Moving Objects. In IDEAS, 2002.
6. F. Korn, S. Muthukrishnan, and D. Srivastava: Reverse Nearest Neighbor Aggregates Over Data Streams. In VLDB,2002.
7. A. Singh, H. Ferhatosmanoglu, and A. S. Tosun: High Dimensional Reverse Nearest Neighbor Queries. In CIKM, 2003.
8. Y. Tao, D. Papadias, and X. Lian: Reverse kNN Search in Arbitrary Dimensionality. In VLDB, 2004.
9. T. Xia, D. Zhang, E. Kanoulas, and Y. Du: On Computing Top-t Most Inuential Spatial Sites. In VLDB, 2005.
10. M.Yiu and N. Mamoulis: Reverse Nearest Neighbors Search in Ad-hoc Subspaces. In ICDE, 2006.
11. H. V. Jagadish, B. C. Ooi, Q. H. Vu, R. Zhang, and A. Zhou: VBI-Tree: A Peer-to-Peer Framework for Supporting Multi-Dimensional Indexing Schemes. In ICDE, 2006.
12. A. Mondal, Yilifu, and M. Kitsuregawa: P2PR-tree: An R-tree-based Spatial Index for Peer-to-Peer Environments. In EDBT, 2004.
13. M. Demirbas and H. Ferhatosmanoglu: Peer-to-peer spatial queries in sensor networks. In P2P, 2003.
14. C. Tang, Z. Xu, and S. Dwarkadas: Peer-to-peer information retrieval using self-organizing semantic overlay networks. In SIGCOMM, 2003.
15. J. Lee, H. Lee, S. Kang, S. Choe, and J. Song: CISS: An efficient object clustering framework for DHT-based peer-to-peer applications. In DBISP2P, 2004.
16. C. Zhang, A. Krishnamurthy, and R. Y.Wang: Skipindex: Towards a scalable peer-to-peer index service for high dimensional data. Technical report, Princeton University, 2004.
17. Y. Shu, K.-L. Tan, and A. Zhou: Adapting the content native space for load balanced. In DBISP2P, 2004.

18. A. Guttman: R-trees: a dynamic index structure for spatial searching. In ACM SIGMOD International Conference on Management of Data, 1984.
19. N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger: The R*-Tree: an efficient and robust access method for points and rectangles. In ACM SIGMOD International Conference on Management of Data, 1990.
20. T. Sellis, N. Roussopoulos, and C. Faloutsos: The R+-tree: a dynamic index for multi-dimensional objects. In VLDB, 1987.
21. S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker: A scalable content-addressable network. In ACM Annual Conference of the Special Interest Group on Data Communication, 2001.
22. J. L. Bentley: Multidimensional binary search trees used for associative searching. Communications of the ACM, 18(9):509–517, Sep 1975.
23. K. Hinrichs and J. Nievergelt: The grid file: A data structure designed to support proximity queries on spatial objects. In Proceedings of the International Workshop on Graphtheoretic Concepts in Computer Science, 1983.
24. S. Berchtold, D. A. Keim, and H.-P. Kriegel: The X-tree: An index structure for high-dimensional data. In VLDB, 1996.
25. D. A. White and R. Jain: Similarity indexing with the SStree. In ICDE, 1996.
26. P. Ciaccia, M. Patella, and P. Zezula: M-tree: An efficient access method for similarity search in metric spaces. In VLDB, 1997.
27. O. D. Sahin, A. Gupta, D. Agrawal, and A. El Abbadi: A peer-to-peer framework for caching range queries. In ICDE, 2004.
28. J. Aspnes and G. Shah: Skip graphs. In Annual ACM-SIAM Symposium on Discrete Algorithms, 2003.
29. M. Schlosser, M. Sintek, S. Decker, and W. Nejdl: HyperCup-Hypercubes, Ontologies and efficient search on P2P networks. In International workshop on agents and peer-to-peer computing, 2002.