Peer-to-Peer Similarity Search based on M-Tree Indexing

Akrivi Vlachou[1]*, Christos Doulkeridis[1]*, and Yannis Kotidis[2]

[1]Dept.of Computer and Information Science, Norwegian University of Science and Technology
[2]Dept.of Informatics, Athens University of Economics and Business
{vlachou,cdoulk}@idi.ntnu.no, kotidis@aueb.gr

Abstract. Similarity search in metric spaces has several important applications both in centralized and distributed environments. In centralized applications, such as similarity-based image retrieval, usually a server indexes its data with a state-of-the-art centralized metric indexing technique, such as the M-Tree. In this paper, we propose a framework for distributed similarity search, where each participating peer stores its own data autonomously, under the assumption that data is indexed locally by peers using M-Trees. In order to support scalability and efficiency of search, we adopt a super-peer architecture, where super-peers are responsible for query routing. We propose the construction of metric routing indices suitable for distributed similarity search in metric spaces. We study the performance of the proposed framework using both synthetic and real data.

1 Introduction

Similarity search in metric spaces has received significant attention in centralized settings [1, 6], but also recently in decentralized environments [3, 5, 8]. A prominent application is distributed search for multimedia content, such as images, video or plain text. Existing approaches for P2P metric-based similarity search mainly rely on a structured P2P overlay, which is used to intentionally store objects to peers [5, 8]. The aim is to achieve high parallelism and share the high processing cost over a set of cooperative computers. In contrast, in this paper we focus on the scenario of autonomous peers that store multimedia content and collaborate in order to process similarity queries over distributed data. A P2P architecture for image retrieval was proposed in [9]. In more details, content providers are simple peers that keep multimedia content, usually generated on their own. Each peer joins the collaborative search engine, by connecting to one of the information brokers that act as super-peers, using the basic bootstrapping protocol. In this scenario the super-peers are responsible for the execution of the queries. In such a distributed search engine, the objective is to find all objects that are similar to a given query object, such as a digital image or a text document. Objects are represented in a high dimensional feature space and a metric distance function defines the similarity of two objects. One of the most commonly used centralized indexing techniques for searching in metric spaces is the M-Tree [2] that consists of a hierarchy of hyper-spheres.

* This work was carried out during the tenure of an ERCIM "Alain Bensoussan" Fellowship Programme.

In this paper, we propose a distributed search mechanism that relies on a super-peer architecture, assuming that cooperative peers store and index their data using an M-Tree in an autonomous manner. Each peer connects to a super-peer and publishes the set of hyper-spheres stored at the root of its M-Tree to its super-peer, as a summarization of the stored data. The super-peers store the collected hyper-spheres using an M-Tree index, in order to direct queries only to relevant peers efficiently, thus establishing a *peer selection mechanism*. Capitalizing on their local metric index structures, super-peers exchange summary information to construct metric-based routing indices, which improve the performance of query routing significantly. Then, given a range query, this *super-peer selection mechanism* enables efficient query routing only to that subset of super-peers that are responsible for peers with relevant query results.

Following the same spirit, in SIMPEER [3], P2P metric-based indexing is supported using the iDistance [7] technique. An extension of SIMPEER for recall-based range queries is presented in [4]. In contrast, this paper provides an alternative technique for similarity search in metric spaces, based on a popular metric index (M-Tree) for data access both on peers and super-peers.

2 Preliminaries

We assume an unstructured P2P network that consists of N_p peers. Some peers have special roles, due to their enhanced features, such as availability, stability, storage capability and bandwidth capacity. These peers are called super-peers SP_i ($i = 1..N_{sp}$), and they constitute only a small fraction of the peers in the network, i.e. $N_{sp} << N_p$. Peers that join the network connect to one of the super-peers directly. Each super-peer maintains links to peers, based on the value of its degree parameter DEG_p, which is the number of peers that it is connected to. In addition, a super-peer is connected to a limited set of at most DEG_{sp} other super-peers ($DEG_{sp} < DEG_p$). In our system, peers that join the network autonomously store their own data. Each peer P_i holds n_i d-dimensional points, denoted as a set S_i ($1 \leq i \leq N_p$). Assuming horizontal data distribution to the N_p peers, the size of the complete set of points is $n = \sum_{i=1}^{N_p} n_i$ and the dataset S is the union of all peers' datasets S_i ($S = \cup S_i$). Each peer maintains its own data objects, while the d-dimensional points are features extracted from the objects.

Similarity search in metric spaces focuses on supporting queries that retrieve objects similar to a query point, when a metric distance function $dist$ measures the objects' (dis)similarity. More formally, a metric space is a pair $M = (\Delta, dist)$, where Δ is a domain of feature values and $dist$ is a distance function with the following properties: 1) $dist(p, q) > 0$, $q \neq p$ and $dist(p, p) = 0$ (non negativity), 2) $dist(p, q) = dist(q, p)$ (symmetry), and 3) $dist(p, q) \leq dist(p, o) + dist(o, q)$ (triangle inequality). Similarity search in metric spaces involves two different types of queries, namely *range* and *nearest neighbor* queries. In this paper, we focus on range queries since k-NN queries can be transformed to range queries, if the distance of the k-th nearest neighbor is known. Radius estimation techniques for distributed nearest neighbor search have been studied in [3].

The M-Tree [2] is a distance-based indexing method, suitable for disk-based implementation. An M-Tree can be seen as a hierarchy of metric regions, also known as hyper-spheres or balls. More precisely, all the objects being indexed are referenced in the leaf nodes, while an entry in a non-leaf node stores a pointer to a node at the next lower level along with summary information about the objects in the subtree being pointed at. The objects in the internal nodes are database objects that are chosen (during the insertion) as representative points. For a non-leaf node N, the entries are quad-tuples $\{(p, r(p)), D, T\}$, where p is an representative object, $r(p)$ is the corresponding covering radius, D is a distance value, and T is a reference to a child node of N. The basic property is that for all objects o in the subtree rooted at T, we have $dist(p, o) \leq r(p)$. For each non-root node N, let object p' be the parent object, i.e. the object in the entry pointing to N. The distance value stored in D is the distance $dist(p, p')$ between p and the parent object p' of N. These parent distances allow more efficient pruning during search than would otherwise be possible. Similarly, for a leaf node N, the entries consist of pairs of the form (o, D), where o is a data object and D is the distance between o and the parent object of N.

3 Metric-based Routing Indices

In our framework, each peer P_i that connects to a super-peer SP_j publishes a summary of its data, in order to make its content searchable by other peers. In our framework, we take advantage of the existing M-Tree index and each peer P_i publishes to its responsible super-peer SP_j, the hyper-spheres contained in the root of the M-Tree, as a summary of the stored data. This set of hyper-spheres covers all data objects stored at P_i, thus SP_j is able to determine if P_i stores data relevant to a potential range query, by searching for hyper-spheres that overlap with the query. SP_j needs to support efficient retrieval of peer hyper-spheres, and consequently selection of the peers that store relevant data to a similarity query. For this purpose, SP_j inserts the collected hyper-spheres into a local M-Tree, also mentioned as *super-peer M-Tree*.

The remaining challenge is to construct routing indices for processing similarity queries over the entire super-peer network. For this purpose, each super-peer maintains an M-Tree, also called *routing M-Tree*, to store hyper-spheres (collected from other super-peers) that describe the data accessible through each neighbor in the super-peer topology. A super-peer SP_i sends the descriptions of the hyper-spheres contained in the root of the super-peer M-Tree to its neighbors. This message has the following format: $(msgId, \{(p_i, r(p_i))\})$, where $msgId$ is an identifier that is unique for each SP_i, and $\{(p_i, r(p_i))\}$ represents the set of SP_i's hyper-spheres corresponding to the root of SP_i's M-Tree. Each hyper-sphere is defined by a representative object p_i and the corresponding covering radius $r(p_i)$. Each neighboring super-peer SP_j that receives a set of hyper-spheres for the first time performs two operations. First, SP_j stores locally the hyper-spheres in the routing M-Tree and attaches to them the identifier of the neighboring super-peer SP_i, from which the hyper-spheres were received. Second, SP_j propagates the hyper-spheres to all its neighbors, except for the one it received them from (SP_i). Any super-peer SP_k that is contacted by SP_j performs the same operations. However, notice that SP_k stores in its routing M-Tree the identifier of its neighbor SP_j together with the hyper-spheres, and not the identifier of the owner super-peer SP_i.

This construction protocol works also for network topologies that contain cycles. Since hyper-spheres of any super-peer SP_i are accompanied by a unique $msgId$, each recipient super-peer SP_k can perform duplicate elimination, in case SP_i's hyper-spheres are also received from a different network path. Notice that the granularity of the routing information stored at any super-peer is at the level of its neighbors $O(DEG_{sp})$ and not at the level of the network $O(N_{sp})$. Therefore, the constructed routing indices are scalable with network size.

We now elaborate more on the internal structure of nodes in the routing M-Tree. For internal nodes, the routing M-Tree entry is $\{(p, r(p)), D, T\}$, where $(p, r(p))$ is the representative object p and its covering radius $r(p)$, D is the distance to the parent object and T is a reference to a subtree. For leaf nodes, the routing M-Tree entry is $\{(p, r(p), SP(p)), D\}$, where $(p, r(p), SP(p))$ consists of the representative object p, its covering radius $r(p)$, and $SP(p)$ is the neighbor super-peer responsible for the hyper-sphere, whereas D is the distance to the parent object.

4 Metric-based Similarity Search

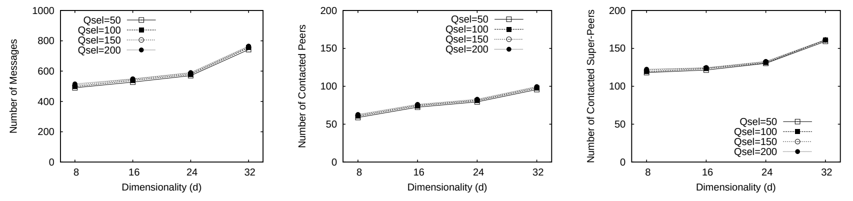
Our framework creates routing M-Trees on each super-peer and supports efficient query processing, in terms of local computation costs, communication costs and overall response time. A query may be posed by any peer P_q and is propagated to the associated super-peer SP_q, which becomes responsible for local query processing and query routing, and finally returns the result set to P_q.

4.1 Super-peer Local Query Processing

Given a range query $R(q, r)$, query processing at SP_i is performed by exploiting the summary information stored in the super-peer M-Tree. The aim is to retrieve the subset of local peers that need to be contacted. The peers that store data enclosed in the range query $R(q, r)$ have to be contacted, since these results are necessary to be retrieved and reported back to SP_q, in order to form the exact and complete result set. Therefore, SP_i uses its super-peer M-Tree to identify hyper-spheres of peers that intersect with the query. Recall that the retrieved hyper-spheres contain the peer identifier of the owner peer. Thus, the subset of peers that can contribute to the query result is determined and the range query is forwarded to the corresponding peers. This enables efficient similarity search over all data stored by peers associated to SP_i, since the query is posed only to peers having data that may appear in the result set, essentially forming an effective *peer selection mechanism* at a super-peer. Each recipient peer processes the query using its local M-Tree, in the traditional way of processing range queries in M-Trees. Consequently, each peer reports its results to SP_i, which in turn is responsible for returning the results to SP_q.

4.2 Query Routing

After having described the local query processing on each super-peer, we proceed to present the details on query routing at super-peer level. Henceforth, we assume that each



(a) Number of messages (b) Contacted peers (c) Contacted super-peers

Fig. 1. Scalability with dimensionality for clustered dataset.

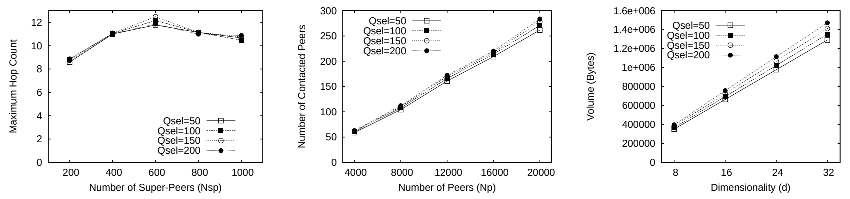
super-peer that receives the query also performs local query processing, as described above. Given a range query $R(q, r)$, the querying super-peer SP_q needs to selectively propagate the query to a fraction of its neighboring super-peers and each intermediate super-peer SP_i that receives the query repeats the same process. The routing algorithm on any super-peer SP_i is based on its routing M-Tree. When a super-peer SP_r receives a range query $R(q, r)$, SP_r uses the routing M-Tree to efficiently retrieve all hyperspheres that have an overlap with the query. Then, the set of neighbor super-peers is determined and the query is forwarded to them only. This forms the *super-peer selection mechanism* that enables routing of queries at super-peer level. Afterwards, the relevant data is collected and sent back to the neighboring super-peer from which the query was received. Finally, SP_q collects all results of its neighboring super-peers and sends the result set back to the peer P_q that posed the query.

5 Experimental Evaluation

In order to evaluate the performance of our approach, we implemented a simulator prototype in Java. For the P2P network topology, we used the GT-ITM topology generator[1] to create well-connected random graphs of N_{sp} peers with a user-specified average connectivity (DEG_{sp}). We used synthetic data collections, in order to study the scalability of our approach. The uniform and clustered datasets are generated as described in [3] and they are horizontally partitioned evenly among the peers by keeping n/N_p=1000 in all setups. Additionally, we employed a real data collection (VEC), which consists of 1M 45-dimensional vectors of color image features. In all cases, we generate 100 queries uniformly distributed and we show the average values. For each query, a peer initiator is randomly selected. Although different metric distance functions can be supported, in this set of experiments we used the Euclidean distance function. We measure the: (i) number of messages, (ii) volume of transferred data, (iii) number of transferred objects, (iv) maximum hop count, (v) number of contacted peers, (vi) number of contacted super-peers, and (vii) response time.

Initially, we focus on the case of clustered dataset. We use a default setup of: N_{sp}=200, N_p=4000, DEG_{sp}=4, n=4M, and the selectivity of range queries ranges from 50 to 200 objects. We study the effect of increasing dimensionality d to our approach. In Fig. 1(a), the number of messages required for searching increases when

[1] Available at: http://www.cc.gatech.edu/projects/gtitm/



(a) Scalability with N_{sp} (b) Scalability with N_p (c) Uniform data

Fig. 2. Scalability for clustered and uniform dataset.

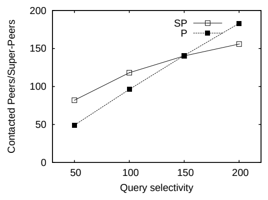
the dimensionality increases. Then, in Fig. 1(b) and 1(c), we measure the number of contacted peers and super-peers respectively. Although the number of super-peers that process the query is between 120 and 150, the number of peers is much lower, ranging from 60 to 100 peers.

In the following, we study the scalability of our approach with respect to the network parameters, by fixing d=8. For this purpose, we increase the number of super-peers N_{sp} (Fig. 2(a)) and peers N_p (Fig. 2(b)). We observe that the maximum hop count increases only slightly, always remaining below 12, when the number of super-peers is increased by a factor of 5. On the other hand, in Fig. 2(b), the increasing number of peers only affects the number of contacted peers, however the increase is only marginal compared to the network size.

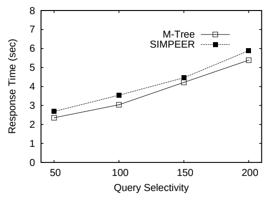
In Fig. 2(c), we examine the case of uniform data. Clearly, this is a hard case for our approach, as a query may in worst case have to contact all peers, in order to retrieve the correct results. This actually occurs in our experiments, causing also a large number of messages to be sent. We show the volume of transferred data in Fig. 2(c). Compared to the case of the clustered dataset, the total volume transferred increases by a factor of 3-4. However, when the maximum hop count is measured, this value is small (equal to 6, for d=8-32), even smaller than in the case of clustered dataset, since the probability of finding the results in smaller distance increases.

In addition, we evaluate our approach using the real dataset (VEC). We used a network of 200 super-peers and 1000 peers, thus each peer stores 1000 data points. In Fig. 3(a), the number of contacted peers and super-peers are depicted for increasing query selectivity from 50 to 200 points. The results are comparable to the case of the clustered dataset, but slightly worse, as the VEC dataset is not clustered. However, notice that the absolute numbers are comparable to the results obtained using the synthetic dataset, which is a strong argument in favor of the feasibility of our approach.

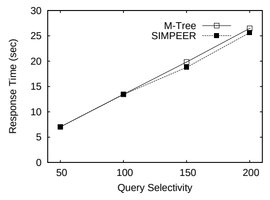
Finally, we study the comparative performance of the proposed framework to SIMPEER [3]. We performed a set of experiments using both approaches, assuming a modest 4KB/sec as network transfer rate. In the case of the uniform dataset, our framework outperforms SIMPEER in terms of response time, as depicted in Fig. 3(b). In contrast, when a clustered dataset is used, SIMPEER is marginally better than our framework, as shown in Fig. 3(c). For the clustered dataset, SIMPEER is able to accurately discover the underlying clusters in the data, resulting in better performance. When the data distribution is uniform, our framework based on M-Trees is more efficient than SIM-



(a) Real data



(b) Uniform data



(c) Clustered data

Fig. 3. Real data and comparison to SIMPEER in terms of response time.

PEER, since the performance of our metric-based routing indices is not influenced by the absence of a clustering structure in the data.

6 Conclusions

Similarity search in metric spaces has several applications, such as image retrieval. In such applications that require similarity search in metric spaces, usually a server indexes its data with a state-of-the-art centralized metric indexing technique, such as the M-Tree. In this paper, we study the challenging problem of supporting efficient similarity queries over distributed data in a P2P system. The experimental results show that our approach performs efficiently in all cases, while the performance of our framework scales with all network and dataset parameters.

References

1. E. Chavez, G. Navarro, R. Baeza-Yates, and J. L. Marroquin. Searching in metric spaces. *ACM Computing Surveys (CSUR)*, 33(3):273–321, 2001.
2. P. Ciaccia, M. Patella, and P. Zezula. M-tree: An efficient access method for similarity search in metric spaces. In *Proc. of VLDB*, pages 426–435, 1997.
3. C. Doulkeridis, A. Vlachou, Y. Kotidis, and M. Vazirgiannis. Peer-to-peer similarity search in metric spaces. In *Proc. of VLDB*, pages 986–997, 2007.
4. C. Doulkeridis, A. Vlachou, Y. Kotidis, and M. Vazirgiannis. Efficient range query processing in metric spaces over highly distributed data. *Distributed and Parallel Databases*, 26(2-3):155–180, 2009.
5. F. Falchi, C. Gennaro, and P. Zezula. A content-addressable network for similarity search in metric spaces. In *Proc. of DBISP2P*, pages 126–137, 2005.
6. G. R. Hjaltason and H. Samet. Index-driven similarity search in metric spaces. *ACM Transactions on Database Systems (TODS)*, 28(4):517–580, 2003.
7. H. V. Jagadish, B. C. Ooi, K.-L. Tan, C. Yu, and R. Zhang. iDistance: An adaptive B$^+$-tree based indexing method for nearest neighbor search. *ACM Transactions on Database Systems (TODS)*, 30(2):364–397, June 2005.
8. D. Novak and P. Zezula. M-Chord: a scalable distributed similarity search structure. In *Proc. of InfoScale*, page 19, 2006.
9. A. Vlachou, C. Doulkeridis, D. Mavroeidis, and M. Vazirgiannis. Designing a peer-to-peer architecture for distributed image retrieval. In *Adaptive Multimedia Retrieval*, pages 182–195, 2007.