



# F I M U

---

Faculty of Informatics  
Masaryk University Brno

## Adaptive Approximate Similarity Searching through Metric Social Networks

by

Jan Sedmidubský  
Stanislav Bartoň  
Vlastislav Dohnal  
Pavel Zezula

FI MU Report Series

FIMU-RS-2007-06

---

Copyright © 2007, FI MU

November 2007

**Copyright © 2007, Faculty of Informatics, Masaryk University.  
All rights reserved.**

**Reproduction of all or part of this work  
is permitted for educational or research use  
on condition that this copyright notice is  
included in any copy.**

**Publications in the FI MU Report Series are in general accessible  
via WWW:**

`http://www.fi.muni.cz/reports/`

**Further information can be obtained by contacting:**

**Faculty of Informatics  
Masaryk University  
Botanická 68a  
602 00 Brno  
Czech Republic**

# Adaptive Approximate Similarity Searching through Metric Social Networks

Jan Sedmidubský

Masaryk University, Brno, Czech Republic

xsedmid@fi.muni.cz

Stanislav Bartoň

Masaryk University, Brno, Czech Republic

xbarton@fi.muni.cz

Vlastislav Dohnal

Masaryk University, Brno, Czech Republic

dohnal@fi.muni.cz

Pavel Zezula

Masaryk University, Brno, Czech Republic

zezula@fi.muni.cz

November 16, 2007

## Abstract

Exploiting the concepts of social networking represents a novel approach to the approximate similarity query processing. We present an unstructured and dynamic P2P environment in which a metric social network is built. Social communities of peers giving similar results to specific queries are established and such ties are exploited for answering future queries. Based on the universal law of generalization, a new query forwarding algorithm is introduced and evaluated. The same principle is used to manage query histories of individual peers with the possibility to tune the tradeoff between the extent of the history and the level of the query-answer approximation. All proposed algorithms are tested on real data and medium-sized P2P networks consisting of tens of computers.

# 1 Introduction

The area of similarity searching is a very hot topic for both research and commercial applications. Current data processing applications use data with considerably less structure and pose much less precise queries than traditional database systems. Examples are multimedia data like images or videos that offer query-by-example search, product catalogs that provide users with preference-based search, scientific data gathered from observations or experimental analyses such as biochemical and medical data, or XML documents that come from heterogeneous data sources on the Web or in intranets and thus does not exhibit a global schema. Such data collections can neither be ordered in a canonical manner nor meaningfully searched by precise database queries that would return exact matches.

This novel situation is what has given rise to similarity searching, also referred to as content-based or similarity retrieval. The most general approach to similarity search, still allowing construction of index structures, is modeled in metric space. Many index structures were developed and surveyed recently [29, 21]. However, the current experience with centralized methods [13] reveals a strong correlation between the dataset size and search costs. Thus, the ability of centralized indexes to maintain a reasonable query response time when the dataset multiplies in size, i.e., the *scalability*, is limited. The latest efforts in the area of similarity searching focus on the design of distributed access structures which exploit more computational and storage resources [4, 14, 6, 5, 25]. Current trends are optimizing and tuning the well-known distributed structures towards better utilization of the available resources.

Another approach to design the access structure suitable for large scale similarity query processing emerges from the notion of *social network*. A social network is a term that is used in sociology since the 1950s and refers to a social structure of people, related either directly or indirectly to each other through a common relation or interest [27]. Using this notion, our approach places the peers of the distributed access structure in the role of people in the social network and creates relationships among them according to the similarity of the particular peer's data. The query processing then represents the search for the community of people – peers related by a common interest – similar data.

Using this data point of view, our designed metric social network is a *cognitive knowledge network* according to the terminology stated in [19]. As for the navigation, social networks exhibit the *small world network topology* [28] where most pairs of nodes are reachable by a short chain of intermediates – usually the average pairwise path length is bound by a polynomial in  $\log n$ . Therefore it is anticipated that a small amount – around six – of transitions will be needed

to find the community of peers holding the answer to a query posed at any of the participating peers in the network. This concept is closely related to *semantic overlay networks* which relate peers in the network semantically. One can view semantics as a way of expressing similarity. Semantic overlays are defined over an existing P2P network, so they can exploit properties of the underlying network, such as navigation. Unlike the usual access structures that retrieve a total answer to each query, the presented approach focuses on retrieving the *substantial part* of the answer yet with *partial costs* compared to the usual query processing.

The paper is structured as follows. In the following two subsections, we summarize the related work and specify the contributions of this paper. In Section 2, we provide the reader with the necessary background. Section 3 contains a concise description of our metric social network, originally proposed in [22], the basic search algorithm along with its analysis and a sketch of experiment trials. We specify a fully-adaptive search algorithm and its theoretical basics in Section 4. Experimental evaluation of the new algorithm and of the query history management procedure is presented in Section 5. Finally, the conclusions are drawn in Section 6.

## 1.1 Related Work

P2P networks were traditionally used for file-sharing (Napster, Gnutella, Freenet). Semantic overlays created upon an ordinary P2P network connect semantically similar peers via relationships in order to route queries efficiently or to speed up files downloading. Tribler [20], Sripanidkulchai et al. [24] are representatives of system for file sharing. Indexing documents by terms or keywords is used in PROSA [7], Jin et al. [15], SempreX [9], 6S [3] and Routing Index [12]. Resource Document Format (RDF) as a model is used in REMINDIN [26], INGA [18] and Grid-Vine [2]. Linari et al. [17] uses language models. PARIS [11] defines a schema of peers' data. A concise survey of semantic overlay networks from different perspectives given as a tutorial is available in [1]. These systems relate peers according to the global knowledge about all data in the peer or according to the results returned by a specific query. Most systems are based on the former approach. The latter alternative allows the ties to be more semantically related, so the navigation can be more effective. REMINDIN, INGA as well as our metric social network use this strategy.

## 1.2 Contributions

The metric social network differs from the other works currently available in two aspects: the relations among peers are defined per query and the metric space used as a model offers greater

extensibility. In this respect, the metric social network can be applied not only to text keywords, but also to images or video, etc.

In this paper, we address the major disadvantages of the metric social network presented in [22], i.e., the poor behavior in larger P2P networks. Below we summarize the contributions of this paper:

- Extending the concept of *confusability* – the confusability of two items originally based only on their distance [23] is not sufficient for social networks because it does not take into account an internal structure of the items (e.g., query radii) or the temporal aspect. The time influences the reliability of items – an ancient piece of social information is not very useful in dynamic environments.
- Adaptive search algorithm – based on the term of confusability, a new search algorithm capable of forming suitable social communities is specified.
- Managing the query history – the metric social network stores the social information with respect to a specific query, so this information (the query history) must be maintained in order to avoid duplicating and aging.

## 2 Background

A very useful, if not necessary, search paradigm is to quantify the *proximity*, *similarity*, or *dissimilarity* of a query object versus the objects stored in a database to be searched. Roughly speaking, objects that are *near* a given query object form the query response set. A useful abstraction for nearness is provided by the mathematical notion of *metric space* [16]. We consider the problem of organizing and searching large datasets from the perspective of *generic* or *arbitrary* metric spaces, sometimes conveniently labeled *distance spaces*.

### 2.1 Metric Space

Suppose a metric space  $\mathcal{M} = (\mathcal{D}, d)$  defined for a domain of objects (or the objects' *keys* or *indexed features*)  $\mathcal{D}$  and a total (distance) function  $d$ . In this metric space, the properties of the function  $d : \mathcal{D} \times \mathcal{D} \mapsto \mathbb{R}$ , sometimes called the *metric space postulates*, are typically characterized as:

$$\forall x, y \in \mathcal{D}, d(x, y) \geq 0 \quad \text{non-negativity,}$$

$$\forall x, y \in \mathcal{D}, d(x, y) = d(y, x) \quad \text{symmetry,}$$

$\forall x, y \in \mathcal{D}, x = y \Leftrightarrow d(x, y) = 0$  identity,

$\forall x, y, z \in \mathcal{D}, d(x, z) \leq d(x, y) + d(y, z)$  triangle inequality.

Examples of distance functions are  $L_p$  metrics (City-block, Euclidean, or maximum distance), the edit distance, or the quadratic-form distance, to name a few.

## 2.2 Similarity Query

Probably the most common type of similarity query is the *range query*  $R(q, r)$ . The query is specified by a query object  $q \in \mathcal{D}$ , with some query radius  $r$  as the distance constraint. The query retrieves all objects found within distance  $r$  from  $q$  from a database  $X \subset \mathcal{D}$ , formally:

$$R(q, r) = \{o \in X, d(o, q) \leq r\}.$$

An alternative way to search for similar objects is to use *nearest neighbor queries*. Specifically,  $kNN(q)$  query retrieves the  $k$  nearest neighbors of the object  $q$ . If the collection to be searched consists of fewer than  $k$  objects, the query returns the whole database. Formally, the response set can be defined as follows:

$$kNN(q) = \{R \subseteq X, |R| = k \wedge \forall x \in R, y \in X - R : d(q, x) \leq d(q, y)\}$$

## 3 Metric Social Network

In this section, we give a short overview of the architecture of the metric social network proposed in [22]. This network operates in an ordinary peer-to-peer environment. The peers of the network are capable of storing their own data and querying other network peers. Interconnection of peers is based on the query-answer paradigm, i.e., the relationships between peers are formed by querying some peers and analyzing the results obtained from them.

### 3.1 Architecture

In the metric social network, each peer organizes a piece of data, can pose similarity queries, and returns answers to the queries. In addition, each peer maintains a list of queries it has asked or answered, called *a query history*. This represents the peer's knowledge about the network and is exploited by search algorithms. Every query in the query history has associated a list of peers that participated in the query answering, which forms relationships among peers with respect to

the particular query. Let assume the peer  $P_{start}$  poses a query  $Q$ . A search algorithm exploits the query history to locate the most similar query (a template query)  $Q_t$  and forwards the query  $Q$  to all peers in the list associated with  $Q_t$ . The contacted peers then evaluate the query on their data and return the partial responses  $A_{P_i}(Q)$ . The final answer is  $A(Q) = \bigcup_{i=1}^n A_{P_i}(Q)$  where  $n$  denotes the total number of peers that participated on the answering. In general, a peer to which  $Q$  was forwarded, can also inspect the query history for better template queries and can forward  $Q$  further. In this respect, the relationships defined by the lists in the query history can be observed as communities of peers that share similar data relevant to the particular query. So, the process of searching can be perceived as locating the best community which is then contacted in order to obtain the query results.

A network peer  $P$  is  $P = (X, H)$  where  $X = \{o_1, \dots, o_l\}$  represents a piece of data and  $H = \{h_1, \dots, h_m\}$ ,  $h_i = (Q, tm, L_P^{Acq}(Q), L_P^{Fri}(Q))$  represents the query history, where each item has the query  $Q$  with the timestamp  $tm$  when it was issued and the pair of ordered lists of identified acquaintances and friends regarding the query  $Q$ .

In the following, we define the quality of answer which is consequently used to define the terms of friends and acquaintances.

### 3.2 Measuring Quality

To distinguish which peer answered better, the *quality* is measured by defining a *quality measuring function*  $Qual(A_{P_i}(Q))$ . It returns a quality object  $\tau_i$  that represents the quality of the peer's answer. Since this object is not necessarily a number, we also define a function to compare two quality objects:

$$\text{compare}_{\text{qual}}(\tau_1, \tau_2) = \begin{cases} -1 & \tau_1 \text{ is better than } \tau_2 \\ 0 & \tau_1 \text{ is the same as } \tau_2 \\ 1 & \tau_1 \text{ is worse than } \tau_2 \end{cases}$$

The quality of the total answer is determined by applying the quality measuring function to  $A(Q)$ . An ordering  $\preceq$  which we call the  $\tau$ -ordering is defined on the peers' answers  $A_{P_1}(Q), \dots, A_{P_n}(Q)$  in  $A(Q)$  according to their qualities. The sequence of peers' answers is ordered by decreasing quality when the following holds:

$$i_1 \dots i_n : \tau_{i_a} \preceq \tau_{i_b} \Leftrightarrow a < b \wedge \text{compare}_{\text{qual}}(\tau_{i_a}, \tau_{i_b}) \neq 1$$

Intuitively, the  $\tau$ -ordering orders the peers by their ability to answer the particular query  $Q$ .

### 3.3 Acquaintances and Friends

As we mentioned before, we distinguish two relationships in the metric social network. Firstly, the friend relationship represents the similarity of nodes – two nodes give a similar answer to same query. Secondly, the acquaintance relationship denotes that the target of the relationship took part in the answer passed to the recipient. Acquaintances play an important role in searching because they can be viewed as links to different communities. In the following, we will use the answer size in objects as the measure of quality, i.e.,  $Qual(A_P(Q)) = |A_P(Q)|$ .

A set of acquaintances for a given query  $Q$  is defined as a set of participating peers in the total answer  $Acq(Q) = \{P, A_P(Q) \neq \emptyset\}$ . Friends are identified in the set of acquaintances as peers that contributed to the answer with a significant partial answer:  $Fri(Q) = \{P, |A_P(Q)| > c \cdot |A(Q)|\}$ , where  $c$  is a positive-value constant.

Every peer  $P$  that is identified as a friend or has initiated the query stores the query  $Q$  with the query-issuing timestamp  $t_m$  and the lists of acquaintances and friends defined as follows (both are  $\tau$ -ordered):

$$L_P^{Acq}(Q) = \begin{cases} \{(P_i, \tau_i) | P_i \in Acq(Q) \wedge \tau_i = Qual(A_{P_i}(Q))\} & \text{if } P \in Fri(Q) \cup \{P_{start}\} \\ \emptyset & \text{otherwise} \end{cases}$$

$$L_P^{Fri}(Q) = \begin{cases} \{(P_i, \tau_i) | P_i \in Fri(Q) \wedge \tau_i = Qual(A_{P_i}(Q))\} & \text{if } P \in Fri(Q) \\ \emptyset & \text{otherwise} \end{cases}$$

In other words,  $P_{start}$  remembers just the list of acquaintances (i.e., no friends) recognized unless it appears among the friends.

### 3.4 Search Algorithm

The query processing using the social network follows the common world concepts for searching. Basically, the best acquaintance (peer) regarding the particular subject is located and then all its friends are contacted to return their part of the answer to  $P_{start}$ .

Initially, the  $P_{start}$  goes through its query history and finds the *template query* (the most similar query)  $Q_t$  to the query  $Q$  that it is processing now. In our setting, we express the similarity of two queries as the distance between their query objects. By exploiting the list of acquaintances of  $Q_t$ , we obtain the best acquaintance to which the query  $Q$  is forwarded. This concept is formalized in Algorithm 1. The process of the query forwarding can be repeated more times to locate the peer that is most promising to hold the searched data. At each peer, a different query

---

**Algorithm 1** Unidirectional Query Forwarding Algorithm *forwardQuery*

---

Input:  $P_{start}$ , contacted peer  $P$ , query  $Q$ , last peer's quality  $\tau_{prev}$

- 1: get entry  $E = (Q_t, tm, L_p^{Acq}(Q_t), L_p^{Fri}(Q_t))$  from the query history  $H$  with  $Q_t$  most similar to  $Q$
  - 2: get  $(P_t, \tau_t)$  corresponding to the best acquaintance from  $L_p^{Acq}(Q_t)$
  - 3: **if**  $compare_{qual}(\tau_t, \tau_{prev}) < 0$  **then**
  - 4:    $forwardQuery(P_{start}, P_t, Q, \tau_t)$
  - 5: **else**
  - 6:   **for all**  $(F, \tau_f) \in L_p^{Fri}(Q_t)$  **do**
  - 7:      $answerQuery(F, P_{start}, Q)$
  - 8:   **end for**
  - 9:   get all objects that satisfy  $Q$
  - 10:   send retrieved objects back to  $P_{start}$
  - 11: **end if**
- 

can be retrieved from the query history. The query forwarding stops when the contacted peer's quality ( $\tau_{prev}$  in the algorithm) is better than any of its acquaintances to which it could possibly pass the query. Initially,  $\tau_{prev}$  is set to zero.

When the best acquaintance regarding the particular query is found, it returns its part of the query answer to the querist. Then it looks up in the query history for the template queries and retrieves the sets of friends associated with that queries and forwards  $Q$  to them as described in Algorithm 2. The query is passed also to friends because it is supposed that they hold similar data which will form substantial parts of the query answer  $A(Q)$ . After contacting, the peers return their partial answers  $A_{P_i}(Q)$  to  $P_{start}$ .

**Cost Analysis** The search algorithm is finite because the peer's quality  $\tau_{prev}$  can only be improved. So, the query forwarding converges. In the worst case, the query can be forwarded through all peers in the network. From our experience, the number of forwarding steps is, however, constant and equal to four (for both the small and large networks). This also confirms the validity of the small world phenomenon. By spreading the query over the community formed by friends, the query answer is improved in terms of completeness. Sending the query repetitively to all friends can lead to an infinite loop. This is restrained by not allowing a peer to evaluate the same query twice. In reality, the social communities formed are compact and the number of friend-contacting steps is usually one or two.

---

**Algorithm 2** Friend-of-Friend Simple Query Answering Procedure `answerQuery`

---

Input: current peer  $P$ ,  $P_{start}$ , query  $Q$

- 1:  $S = \{(Q_t, tm, L_p^{Acq}(Q_t), L_p^{Fri}(Q_t)) \mid \text{the query object of } Q_t \text{ satisfies } Q\}$
  - 2: **for all**  $(Q_t, L_p^{Acq}(Q_t), L_p^{Fri}(Q_t)) \in S$  **do**
  - 3:   **for all**  $(F, \tau_f) \in L_p^{Fri}(Q_t)$  **do**
  - 4:     `answerQuery`( $F, P_{start}, Q$ )
  - 5:   **end for**
  - 6: **end for**
  - 7: get all objects that satisfy  $Q$
  - 8: send retrieved objects back to  $P_{start}$
- 

**Experiment Trials** To demonstrate the properties of this search algorithm, we assembled the metric social network consisting of forty-seven peers organizing 100,000 3-D vectors (three values of color histograms of images) compared using the euclidean distance. The initial state of the network was random – in the query history, every peer had five queries with the random number of randomly selected friends (one to fifteen) scored by a random quality. The data was distributed into the peers using the M-tree [10] metric tree structure, where the content of each leaf node was stored in one peer. The M-tree is used to obtain precise and complete query results. It also forms the baseline in terms of costs, i.e., the percentage of peers that participate in the answering. The M-tree access structure is required only to compare the results, so the *recall* of the metric social network can be evaluated. The metric social network does not rely on it and can operate independently.

Figure 1 presents results obtained by executing range queries ( $r = 200$ ) in the metric social network. Average values of recall and costs were measured on a fixed test series of twenty range queries. Between two test series a batch of fifty randomly-picked queries with the same radius was executed. These queries were adapting the social information. The adaptation of the network was naturally turned off during test series. From the results, the reader can observe that the search algorithm is capable of evolving in terms of increasing the recall even if the initial state was completely random, thus, unreliable. The query costs are more or less constant. Their reduction after the 10<sup>th</sup> series is caused by intentionally deleting the initial random state. This proves that the network no longer uses the randomly-generated social information to navigate and answer queries.

We have also run this experiment on a larger network consisting of 150 peers initialized in the very same way. However, the results revealed that this search algorithm is incapable of increasing

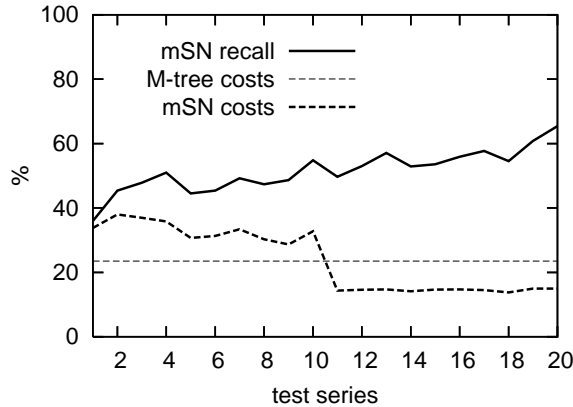


Figure 1: Recall and costs of the social network compared to the M-tree.

the recall. This is caused by the much lower connectivity in this network compared to the smaller one.

Overall, the disadvantages of this search algorithm are twofold. First, the algorithm requires a highly-connected network in order to operate at higher recall values. Second, the peers' query history is not maintained in any way, so obsolete queries (e.g. the initial random state) cannot be superseded and increase the search costs. In the following, we propose a new adaptive algorithm which solves these issues.

## 4 Adaptive Search Algorithm

In this section, we define a new search algorithm which is adaptive in sending the query to more acquaintances considering the closeness of queries retrieved from the query history. In this respect, it allows contacting more acquaintances when good template queries cannot be found.

First, we define confusability on top of which the adaptive search algorithm is built. Second, the adaptive search algorithm is described. Third, the confusability is exploited in a query-history management.

### 4.1 Confusability

In [23], Shepard proposed a "Universal Law of Generalization" which was further studied and reformed by Chater and Vitanyi [8]. The law states that the probability of confusing two items  $x$  and  $y$  is a negative exponential function of the distance  $d(x, y)$ . By exploiting this, we define the *confusability* of two queries as the weighted compound of distance similarity (following the original proposal), query intersection, and time similarity. The value of confusability controls

the number of template queries used for searching in the adaptive search algorithm (refer to Section 4.2).

$$\text{Confusability}(Q, Q_t) = w_D \cdot D(Q, Q_t) + w_I \cdot I(Q, Q_t) + w_T \cdot T(Q, Q_t)$$

The weights must satisfy:  $0 \leq w_{D,I,T} \leq 1$  and  $w_D + w_I + w_T = 1$ . So, the higher the confusability of the new query  $Q$  and a template query  $Q_t$  is, the better the query  $Q_t$  is for searching. The values of confusability are between zero and one. The individual similarities are defined as follows:

- The distance similarity  $D(Q, Q_t)$  expresses the Shephard's confusability of queries  $Q$  and  $Q_t$  with respect to the distance between their query objects  $q$  and  $q_t$ .

$$D(Q, Q_t) = A \cdot e^{-B \cdot d(q, q_t)}$$

The constant  $A$  is set to one in order to keep the values returned between zero and one. The constant  $B$  depends on the distance function  $d$  and the dataset. For our experiments on 3-D data, we fixed  $B = \frac{1}{200}$  because we required low similarity of two queries with  $r = 100$  touching each other, i.e.,  $D(Q, Q_t) = 0.37$  where  $d(q, q_t) = 200$ . If the distance of 400, the similarity is only 0.13.

- The intersection function  $I(Q, Q_t)$  expresses the similarity in terms of the intersecting area of the corresponding query regions (ball regions in the case of range queries). It is defined as the intersection of the ball regions in 2-D space and normalized using the area covered by  $Q$ .

$$I(Q, Q_t) = \frac{|\text{region}(Q) \cap \text{region}(Q_t)|_{2D}}{|\text{region}(Q)|_{2D}}$$

The main reason for defining it is as follows: a new query with large radius is to be answered; two template queries at approximately the same distance are retrieved, however, the first is smaller than the second. Because the second query has large area with the new query in common, it is more profitable from the searching point of view.

- The function  $T(Q, Q_t)$  expresses the similarity of queries  $Q$  and  $Q_t$  with respect to their time difference. In the query history, each query  $Q_t$  has a timestamp  $tm_{Q_t}$  assigned. It corresponds to the time when the query  $Q_t$  was inserted into the query history. The time can be expressed absolutely (wall-clock time) or relatively (a sequence number).

$$T(Q, Q_t) = \max \left( 0, \frac{tm_{\max} - |tm_Q - tm_{Q_t}|}{tm_{\max}} \right)$$

The constant  $tm_{\max}$  should be set to a reasonable value which will mark archaic queries. For our experiments, we used the relative timestamps and fixed  $tm_{\max}$  to 4,000, i.e., a query is considered obsolete after the system processed next 4,000 queries.

## 4.2 Adaptive Search Algorithm

The adaptive search algorithm we propose follows the same idea as the previous search algorithm. The pseudo-code is available in Algorithm 3 presented in Section 3.4. Its adaptability is wired in an automatic increase or decrease of the flooding factor during query forwarding. In our setting, we defined the most simple variant which, however, turned out to be very effective. We defined a threshold value  $\text{threshold}_{\text{conf}} = 0.9$  and a step value of 0.25 to establish five intervals having the increasing flooding factors assigned:

<b>Confusability</b> $\geq$	0.90	0.65	0.40	0.15	0.00
<b>floodingFactor</b>	1	2	3	4	5

In particular, we retrieve five closest template queries from the query history with respect to the distance between the query objects, i.e.,  $k = 5$ , and compute their confusabilities with respect to the query  $Q$ . Depending on the maximum of the confusabilities, one to five template queries are used for query forwarding. As usual, the best acquaintances of template queries are picked and the query  $Q$  is forwarded to them. Notice that different template queries can have the same best acquaintance, hence the query is forwarded only once. Finally, data satisfying the query  $Q$  are retrieved from friends – the function `answerQuery()` is called (Algorithm 2). The experimental evaluation of properties of the adaptive algorithm is presented in Section 5.

## 4.3 Managing the Query History

The other drawback of the basic search algorithm presented earlier is the lack of any management of the query history on peers. The query history was unlimited in size, which is not usually feasible in production applications. Furthermore, each peer has to search its query history for the most similar queries. Such a process has to be optimal because the query can be forwarded several times, so searching in the query history forms a sequential step in the query evaluation. As a result, the size of the query history should be kept in certain boundaries. The content of query history is important as well since obsolete queries lead to increased query costs, which has been depicted in Figure 1 in case of the random initial state. So, the template queries becoming obsolete should be naturally expelled.

---

**Algorithm 3** Adaptive Query forwarding algorithm `forwardQueryAdapt`

---

Input:  $P_{start}$ , contacted peer  $P$ , query  $Q$ , last peer's quality  $\tau_{prev}$

- 1: get the  $k$  most similar entries  $E$  from the query history  $H$
- 2: **for** each entry  $E = (Q_t, tm, L_p^{Acq}(Q_t), L_p^{Fri}(Q_t))$  retrieved **do**
- 3:   compute  $Confusability(Q, Q_t)$
- 4: **end for**
- 5: order entries by decreasing confusability
- 6:  $conf_{max} =$  maximum confusability among all entries  $E$
- 7:  $n =$ floodingFactor( $conf_{max}$ )
- 8: **for all**  $n$  first entries  $E = (Q_t, tm, L_p^{Acq}(Q_t), L_p^{Fri}(Q_t))$  **do**
- 9:   get  $(P_t, \tau_t)$  corresponding to the best acquaintance from  $L_p^{Acq}(Q_t)$
- 10:   **if**  $compare_{qual}(\tau_t, \tau_{prev}) < 0$  **then**
- 11:     forwardQueryAdapt( $P_{start}, P_t, Q, \tau_t$ )
- 12:   **else**
- 13:     **for all**  $(F, \tau_f) \in L_p^{Fri}(Q_t)$  **do**
- 14:       answerQuery( $F, P_{start}, Q$ )
- 15:     **end for**
- 16:     get all objects that satisfy  $Q$
- 17:     send retrieved objects back to  $P_{start}$
- 18:   **end if**
- 19: **end for**

---

We introduce the query history management procedure which optimizes the content of query history and by tuning its parameters it can control the length of query history too. The procedure is specified in Algorithm 4. This piece of code is executed at  $P_{start}$  after a query  $Q$  has been evaluated. The algorithm decides not to insert the new query  $Q$  into the query history if its confusability to a query  $Q_t$  that is already present in the query history, is high, i.e.  $Confusability(Q, Q_t) \geq threshold_{conf}$ . In other words, the query  $Q_t$  is up-to-date and very close to the new query  $Q$ , so there is no need to update the social information in the network. If the confusability is not very high, the new query  $Q$  is inserted into the query history and some obsolete items can get deleted. The obsolete items are selected among the  $k$  most similar template queries retrieved during the evaluation of search algorithm (Algorithm 3). A template query  $Q_t$  gets replaced if its *replaceability* is higher than a threshold value, i.e.  $Replaceability(Q, Q_t) \geq threshold_{repl}$ .

---

**Algorithm 4** Management of the query history

---

Input: a query  $Q$

- 1: use the  $k$  most similar entries  $E$  retrieved in
  - 2:   forwardQueryAdapt()
  - 3: **for** each entry  $E = (Q_t, tm, L_p^{Acq}(Q_t), L_p^{Fri}(Q_t))$  **do**
  - 4:   compute Confusability( $Q, Q_t$ )
  - 5: **end for**
  - 6:  $conf_{max} =$  maximum confusability among all entries  $E$
  - 7: **if**  $conf_{max} \geq threshold_{conf}$  **then**
  - 8:   exit {no management necessary}
  - 9: **end if**
  - 10: insert a new entry corresponding to  $Q$  into  $H$
  - 11: **for** each entry  $E = (Q_t, tm, L_p^{Acq}(Q_t), L_p^{Fri}(Q_t))$  **do**
  - 12:   compute Replaceability( $Q, Q_t$ )
  - 13:   **if**  $Replaceability(Q, Q_t) \geq threshold_{repl}$  **then**
  - 14:     delete the entry  $E$  from  $H$
  - 15:   **end if**
  - 16: **end for**
- 

The replaceability is defined in the similar way as the confusability. However, for the sake of managing the query history, the time component plays the inverse role:

$$Replaceability(Q, Q_t) = w_D \cdot D(Q, Q_t) + w_I \cdot I(Q, Q_t) + w_T \cdot (1 - T(Q, Q_t))$$

The higher the replaceability, the greater the need to replace the template query  $Q_t$  with  $Q$  is. To put it in a different way, the template query  $Q_t$  should be replaced with  $Q$  if the queries are similar and  $Q_t$  is old. In Section 5, we evaluate this management procedure.

## 5 Experimental Results

In this section, we present an experimental evaluation of the proposed adaptive navigation algorithm for searching in metric data. The experiments have been conducted on two real-life datasets represented by vectors having forty five and three dimensions compared using the euclidean distance, respectively. The 45-D vectors represent extracted color image features. The distribution of the dataset is quite uniform and such a high-dimensional data space is extremely

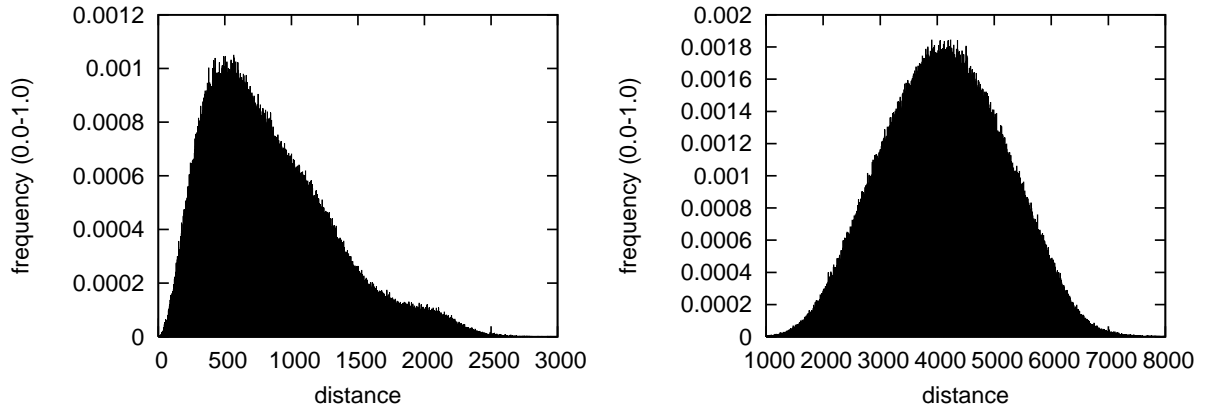


Figure 2: Distance densities of datasets: 3-D vectors (left) and 45-D vectors (right).

sparse. The 3-D vectors were obtained as the three most-important dimensions of the 45-D vectors. The number of vectors in each dataset was 200,000. Figure 2 depicts distance densities of the particular datasets.

## 5.1 Network Initialization

The datasets were distributed over peers using the M-tree index structure [10], where the content of each leaf node was stored in one peer. Besides this, the M-tree was used to obtain precise and complete query results. In particular, the M-tree is created on the provided dataset. Next, the peers are assigned their pieces of data.<sup>1</sup> The initial state of the network is random – in the query history, every peer has five queries with the random number of randomly selected friends (one to fifteen) scored by a random quality.

## 5.2 Methodology

In the individual graphs, we present values of recall and costs obtained by executing test series for one-hundred times. The test series consists of twenty range queries with randomly picked objects and the radii fixed to 100 and 1,500 for the 3-D and 45-D data, respectively. The values of recall and costs are averages over the results of the test series queries. On average, queries with the selected query radii return 0.8% of the dataset. The costs represent the ratio between the number of accessed peers and the number of all peers in the network. The evolution of the metric social network is ensured by executing a batch of 150 random queries having the same radius as the test queries. Remark that any adaptation of the social network information was suspended

<sup>1</sup>For both datasets, the M-tree consists of 150 leaf nodes, i.e., the data objects are distributed among 150 peers.

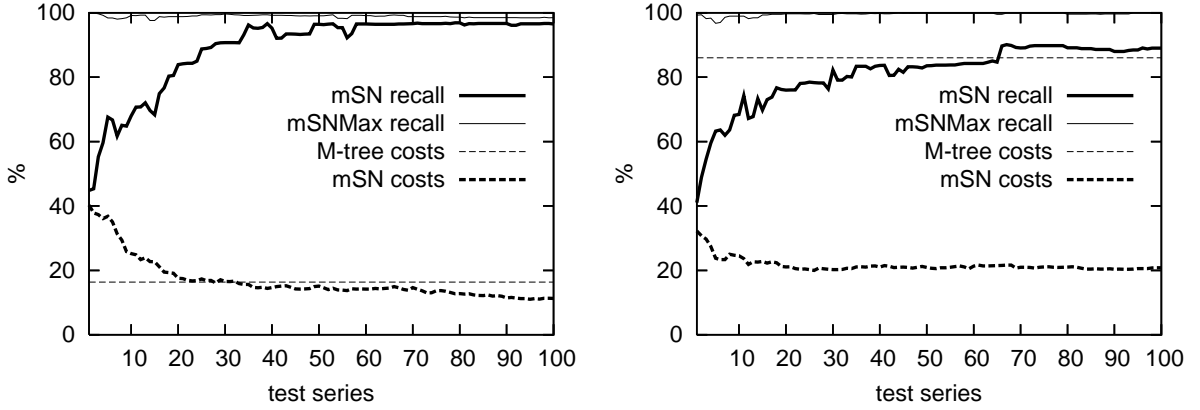


Figure 3: Learning abilities of the metric social network with the *forwardQueryAdapt* navigation algorithm: 3-D vectors (left) and 45-D vectors (right).

during the test series. Moreover, we also present the behavior of the number of template queries stored in each peer’s query history. We provide the reader with the minimum, maximum and average values over all peers.

### 5.3 Recall and Costs

In this section, we present the ability of the metric social network to self-learn towards better query processing when the proposed adaptive navigation algorithm is employed. The results are demonstrated in Figure 3. The recall values are presented for mSN (the metric social network) and mSNMax which corresponds to the recall that could have been obtained if the best peers had been contacted (the same number as mSN did, of course).

For the experiment, we set the weights of the confusability function  $\text{Confusability}(Q, Q_t)$  to  $w_D = 0.77$ ,  $w_I = 0.0$ , and  $w_T = 0.23$ , i.e., we naturally put greater emphasis on the distance but we also use the time similarity to implement aging. We ignore the intersection similarity since the query radii are the same. So, the influence of the intersection similarity was added to the distance similarity. The parameter  $B$  of the distance similarity was set to  $\frac{1}{200}$  and  $\frac{1}{3000}$  for 3-D and 45-D data, respectively. Both the values correspond to approximately the same similarity on both the datasets.

Due to the social information adjustments, the metric social network is improving. In the beginning, the confusability of template queries retrieved from the query history was low, so more queries were used for navigation (as controlled by the *floodingFactor*). After several iterations, the social information was updated and the confusability values increased, so the flooding of the

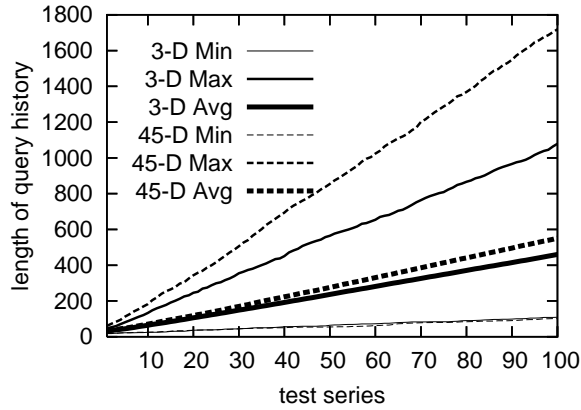


Figure 4: Minimum, maximum and average length of peers’ query histories for the 3-D and 45-D vectors without any query history management.

network was decreased automatically. Finally, after all one hundred iterations of the test series, the recall values reached 95% and 90% for the 3-D and 45-D vectors, respectively, while the costs are still decreasing. This is attributed to the diminished use of template queries containing initial randomly-generated social information. These queries stopped being used due to their aging (the influence of the time similarity of confusability). The reader can observe that the amount of the peers contacted is smaller than the amount of the M-tree leaf nodes in the case of the 3-D data. The gap between those is substantially greater for the 45-D data. This is caused by the sparseness of the data space and the construction principles of the M-tree.

## 5.4 Managing the Query History

In this stage, we study the metric social network from the perspective of peers’ query histories. In Figure 4, their lengths are depicted, namely the minimum, maximum and average over all peers. These results and results in Figure 3 were obtained during the same experimental trial. Without any management of query history, the peers store all queries passed and the peers’ query histories can become overwhelming after a long run-time. For both the datasets, the average length of query history is around 500 after the last iteration of the test series.

By applying the procedure outlined in Algorithm 4, we would like to maintain the content of peers’ query histories and keep them in certain bounds. At the same time, the search costs should decrease because obsolete template queries would be expelled permanently. We used the same weights for the replaceability function  $\text{Replaceability}(Q, Q_t)$  as for the confusability function. The threshold values for confusability and replaceability were 0.9 and 0.75, respectively. We

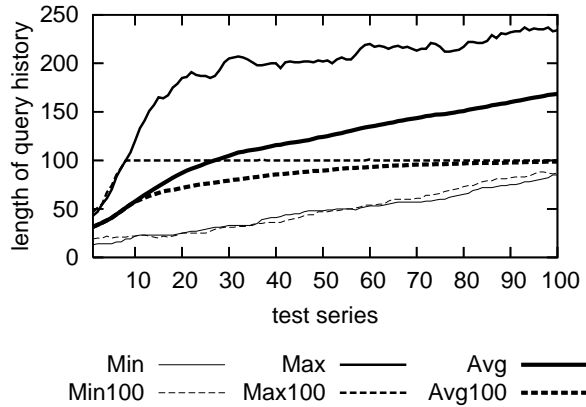


Figure 5: Minimum, maximum and average length of peers’ query histories for the 3-D vectors with the query history management.

present results obtained by experimenting on the 3-D dataset only because the results for the other dataset were very much alike.

In Figure 5, the evolution of lengths of peers’ query histories is depicted by curves labeled with Min, Max and Avg. We can observe that the number of stored queries does not grow as fast as it did in Figure 4. Such a behavior was expected. In early stages (first twenty iterations), the social information was poor, so the peers’ query histories grew rapidly – new queries were inserted in the query history and none were deleted because of low values of both the confusability and replaceability functions. Next, the length of query histories became saturated and the replaceability function instructed the query history management algorithm to supersede some template queries. After all one hundred iterations of the test series, the average length of peers’ query histories was 165 which is nearly three times fewer than without any management, while the quality of the query results (recall) was maintained at the same level, please refer to Figure 6.

In some applications, there is a limited amount of memory which can be allocated for the query history. We tried to model such a situation by posing a hard limit on the length of the query history. If the length of the query history on a peer exceeded one hundred queries, the oldest query was removed until the constraint was met. The development of lengths of peers’ query histories is depicted by curves labeled with Min100, Max100 and Avg100 in Figure 5. Such a hard limit had surprisingly no impact on the recall values. We attribute such a stable behavior to the high adaptability of the proposed algorithm *forwardQueryAdapt* and fine-tunable query history management. Both these algorithms synergically cooperate and increase flooding of the network or storing important queries when necessary.

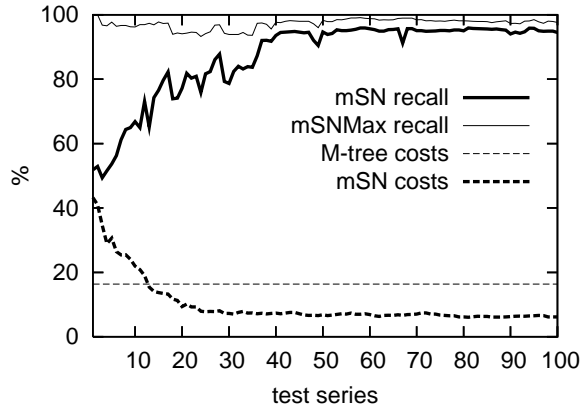


Figure 6: Recall and costs of *forwardQueryAdapt* algorithm with the query history management.

## 6 Concluding Remarks and Future Work

Distributed processing of similarity queries currently attracts a lot of attention because of its inherent capability of solving the issue of data scalability. We have proposed two algorithms which address two drawbacks of the metric social network – namely the navigation algorithm which limited exploration of the network and the ever-growing query history which contained also obsolete items. The principles of both the algorithms exploit so-call *confusability* which is based on the *law of generalization*. The presented experiment trails confirm suitability and auspiciousness of such advances.

In our system, we have no automatic exploration implemented, no background actions are done by peers automatically, and peers do not exchange any profiles about their data but the query results. We did not experiment on an actual peer-to-peer infrastructure where peer disconnections happen frequently due to shutting down the peer or a network failure. Dealing with such environments is the future work.

Another research challenge is to study the behavior of the metric social network when the data is partitioned differently (the M-tree tries to cluster data) or when the data change not only in terms of adding or deleting data items but also in terms of different domains, e.g., music, video, or text. We also plan to verify the metric social network properties on large-scale networks consisting of hundreds or thousands of peers.

### 6.1 Acknowledgements

Partially supported by: the EU IST FP6 project 045128 (SAPIR), the national research project 1ET100300419, the Czech Grant Agency project 201/07/P240.

## References

- [1] K. Aberer and P. Cudré-Mauroux. Semantic overlay networks. In *Proceedings of the 31st International Conference on Very Large Data Bases (VLDB 2005), Trondheim, Norway, August 30 - September 2, 2005*, page 1367. ACM Press, 2005.
- [2] K. Aberer, P. Cudre-Mauroux, M. Hauswirth, and T. van Pelt. GridVine: Building internet-scale semantic overlay networks. In *International Semantic Web Conference (ISWC)*, volume 3298 of *LNCS*, pages 107–121, 2004.
- [3] R. Akavipat, L.-S. Wu, F. Menczer, and A. Maguitman. Emerging semantic communities in peer web search. In *P2PIR '06: Proceedings of the international workshop on Information retrieval in peer-to-peer networks*, pages 1–8, New York, NY, USA, 2006. ACM Press.
- [4] F. Banaei-Kashani and C. Shahabi. SWAM: A family of access methods for similarity-search in peer-to-peer data networks. In *CIKM '04: Proceedings of the Thirteenth ACM conference on Information and knowledge management*, pages 304–313. ACM Press, 2004.
- [5] M. Batko, D. Novak, F. Falchi, and P. Zezula. On scalability of the similarity search in the world of peers. In *Proceedings of First International Conference on Scalable Information Systems (INFOSCALE 2006), Hong Kong, May 30 - June 1*, pages 1–12. ACM Press, 2006.
- [6] M. Bender, S. Michel, P. Triantafillou, G. Weikum, and C. Zimmer. MINERVA: Collaborative P2P search. In *VLDB '05: Proceedings of the 31st international conference on Very large data bases*, pages 1263–1266. VLDB Endowment, 2005.
- [7] V. Carchiolo, M. Malgeri, G. Mangioni, and V. Nicosia. Social behaviours in p2p systems: An efficient algorithm for resource organisation. In *Proceedings of the 15th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE 2006), 26-28 June 2006, Manchester, United Kingdom*, pages 65–72. IEEE Computer Society, 2006.
- [8] N. Chater and P. M. Vitanyi. The generalized universal law of generalization. *Journal of Mathematical Psychology*, 47(3):346–369, 2003.
- [9] H. Chen, H. Jin, and X. Ning. Semantic peer-to-peer overlay for efficient content locating. In *APWeb Workshops*, pages 545–554, 2006.
- [10] P. Ciaccia, M. Patella, and P. Zezula. M-tree: An efficient access method for similarity search in metric spaces. In M. Jarke, M. J. Carey, K. R. Dittrich, F. H. Lochovsky, P. Loucopoulos, and M. A. Jeusfeld, editors, *Proceedings of the 23rd International Conference on Very Large Data Bases (VLDB 1997), Athens, Greece, August 25-29, 1997*, pages 426–435. Morgan Kaufmann, 1997.
- [11] C. Comito, S. Patarin, and D. Talia. A semantic overlay network for p2p schema-based data integration. *ISCC*, 0:88–94, 2006.
- [12] A. Crespo and H. Garcia-Molina. Routing indices for peer-to-peer systems. In *Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS'02)*, page 23, Washington, DC, USA, 2002. IEEE Computer Society.
- [13] V. Dohnal, C. Gennaro, P. Savino, and P. Zezula. D-index: Distance searching index for metric data sets. *Multimedia Tools and Applications*, 21(1):9–33, 2003.

- [14] P. Ganesan, B. Yang, and H. Garcia-Molina. One torus to rule them all: Multi-dimensional queries in P2P systems. In *WebDB '04: Proceedings of the 7th International Workshop on the Web and Databases*, pages 19–24, New York, NY, USA, 2004. ACM Press.
- [15] H. Jin, X. Ning, H. Chen, and Z. Yin. Efficient query routing for information retrieval in semantic overlays. In *Proceedings of the 2006 ACM Symposium on Applied Computing (SAC 2006), Dijon, France, April 23-27, 2006*, pages 1669–1673. ACM Press, 2006.
- [16] J. L. Kelly. *General Topology*. D. Van Nostrand, New York, 1955.
- [17] A. Linari and G. Weikum. Efficient peer-to-peer semantic overlay networks based on statistical language models. In *P2PIR '06: Proceedings of the international workshop on Information retrieval in peer-to-peer networks*, pages 9–16, New York, NY, USA, 2006. ACM Press.
- [18] A. Löser, S. Staab, and C. Tempich. Semantic methods for p2p query routing. In *MATES*, pages 15–26, 2005.
- [19] P. R. Monge and N. S. Contractor. *Theories of Communication Networks*. Oxford University Press, April 2003.
- [20] J. A. Pouwelse, P. Garbacki, J. Wang, A. Bakker, J. Yang, and A. Iosup. Tribler: A social-based peer-to-peer system. In *Proceedings of the 5th International P2P conference (IPTPS 2006)*, pages 1–6, 2006.
- [21] H. Samet. *Foundations of Multidimensional And Metric Data Structures*. The Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, 2006.
- [22] J. Sedmidubský, S. Bartoň, V. Dohnal, and P. Zezula. Querying similarity in metric social networks. In *Proceedings of the 1st International Conference on Network-Based Information Systems (NBIS 2007), Regensburg, Germany, September 3-7, 2007*, volume 4658 of *Lecture Notes in Computer Science*, pages 278–287. Springer, 2007.
- [23] R. N. Shepard. Toward a universal law of generalization for psychological science. *Science*, 237(4820):1317–1323, 1987.
- [24] K. Sripanidkulchai, B. M. Maggs, and H. Zhang. Efficient content location using interest-based locality in peer-to-peer systems. In *INFOCOM*, pages 1–11, 2003.
- [25] E. Tanin, D. Nayar, and H. Samet. An efficient nearest neighbor algorithm for P2P settings. In *Proceedings of the 2005 national conference on Digital government research*, pages 21–28. Digital Government Research Center, 2005.
- [26] C. Tempich, S. Staab, and A. Wranik. Remindin’: Semantic query routing in peer-to-peer networks based on social metaphors. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 640–649, New York, NY, USA, 2004. ACM Press.
- [27] S. Wasserman, K. Faust, and D. Iacobucci. *Social Network Analysis : Methods and Applications (Structural Analysis in the Social Sciences)*. Cambridge University Press, November 1994.
- [28] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, June 1998.

- [29] P. Zezula, G. Amato, V. Dohnal, and M. Batko. *Similarity Search: The Metric Space Approach*, volume 32 of *Advances in Database Systems*. Springer, 2005.