Similarity Searching in Structured and Unstructured P2P Networks

Vlastislav Dohnal and Pavel Zezula

Faculty of Informatics, Masaryk University,
Botanicka 68a, 602 00 Brno, Czech Republic
{dohnal,zezula}@fi.muni.cz

Abstract. The exponential growth of digital data in contemporary computer networks induces a lot of scalability, resilience, and survivability issues. At the same time, the increasing complexity of digital data makes the task of similarity searching that is inherently difficult, more and more important. In this paper, we report on the Multi Feature Indexing Network, MUFIN, which is an extensible, scalable, and infrastructure independent similarity search engine. It is able to achieve high performance and guarantee quality of service by applying structured Peer-to-Peer networks. On the other hand, its unstructured version based on self-organizing principles is extremely robust and able to operate in very volatile environments. To exemplify MUFIN's properties, an on-line demo is available for public use.

Keywords: similarity searching, structured peer-to-peer network, unstructured peer-to-peer network, self-organizing system, metric space, scalability, resilience to failures, performance evaluation.

1 Introduction

Similarity is a central notion throughout human lives. In perception, the similarity between sets of visual or auditory stimuli influences the way in which they are grouped. In speech recognition, the similarity between different phonemes determines how confusable they are. In classification, the category of a new instance may be influenced by the similarity of the new instance to past instances or to a stored prototype. In memory, it has been suggested that retrieval of a cue depends on similarity of past memory traces to the representation of the cue. Since almost everything that we see, hear, read, write and measure is, or very soon will be, available in a digital form, computer systems must support similarity. But the growth of the amount of digital material distributed in large-scale wired and wireless networks is posing another big challenge. The exponential increase of data volume makes the performance, configuration, cross-layer approaches, scalability, resilience and survivability an important matter of concern. However, the core ability of future data processing systems should be developed around effective and efficient similarity management of very large and growing collections of data.

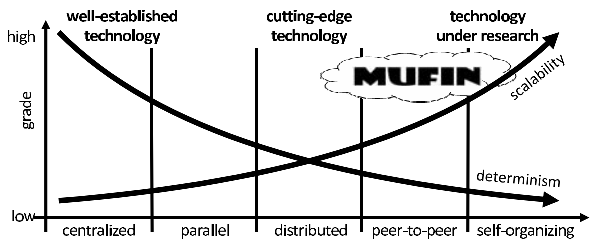

Fig. 1. Trade-off between scalability and determinism in system control

As Figure 1 outlines, we believe that a future search system will be created upon the divergence of scale and determinism. The scalability will be more and more important with respect to the data volume, number of users, query execution response time, number of different query types produced by digitization and content enrichment techniques, as well as the multi-modal approach to querying. On the other hand, the determinism in answering queries, i.e. providing always the same answers to the same queries, will be substituted by satisfactory results or even recommendations. Queries will also be much more personalized and influenced by context and executed on hardware most suited to the given workload. In any case, the exact match will be more and more often accompanied by extensive use of similarity searching.

In this paper, we present our Multi-Feature Indexing Network initiative and explain how our approach can contribute to the quality of service and robustness objectives of future similarity search systems. In particular, we give an architectural view of Multi-Feature Indexing Network in Section 2. Two instances of MUFIN defined as structured Peer-to-Peer networks are summarized in Section 3. Whereas, a system operating as an unstructured Peer-to-Peer system with self-organizing abilities is described in Section 4. Both these sections are accompanied with a sketch of experimental trials showing their properties. In Section 5, a demonstration application for image content-based retrieval is given. As for future applications, there are two examples of searching by biometric characteristics. The paper concludes in Section 6.

2 Multi-Feature Indexing Network

In this section, we present and demonstrate capabilities of the Multi-Feature Indexing Network, so-called MUFIN [1]. From a general point of view as shown in Figure 2, the search problem has three dimensions: (i) data and query types, (ii) index structures and search algorithms, and (iii) infrastructure to run a system on. MUFIN adopts the *metric space* model of similarity. Its indexing and searching mechanisms are based on the concept of *structured* and *unstructured* Peer-to-Peer (P2P) networks, which makes the approach highly scalable and independent of the specific hardware infrastructure.

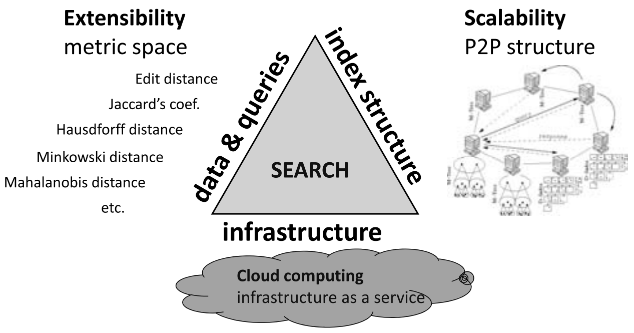

Fig. 2. Basic concept of MUFIN [2]

2.1 Modeling Similarity

The metric space model of similarity has already proved to be a very powerful concept for expressing many different forms of similarity of vectors, strings, sets and other data types. Most of the available technologies for processing metric data have been summarized in a recent book [3].

A *metric space* $\mathcal{M} = (\mathcal{D}, d)$ is defined for a *domain* of objects (or extracted features) $\mathcal{D}$ and a total function d that evaluates *distance* between a pair of objects. The properties of this function are: non-negativity, symmetry and triangle inequality. The distance expresses *dissimilarity* between two objects. Examples of distance functions are L_p metrics (City-block (L_1) or Euclidean (L_2) distance), the edit distance, or the quadratic-form distance. Whereas examples of objects are a color histogram extracted from an image and stored as a vector, or a shape of hand expressed as a polygon.

There are two basic types of similarity queries: *range query* and *k-nearest neighbors query*. The range query $R(q, r)$ is specified by a query object $q \in \mathcal{D}$ and a query radius r. From a database $X \subset \mathcal{D}$, the query retrieves all objects found within the distance r from q. The definition is as follows:

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

Whenever we want to search for similar objects using a range search, we must specify the maximum distance for objects to qualify. But it can be difficult to specify it without some knowledge of the data and the distance function. An alternative way to search for similar objects is to use the k-nearest neighbor query $kNN(q)$. It retrieves the k nearest neighbors of the object q. Formally, the response set can be defined as follows:

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

2.2 Architecture

MUFIN, schematically depicted in Figure 3, has a four-tier architecture. The lowest tier is represented by a computer network and its hardware infrastructure

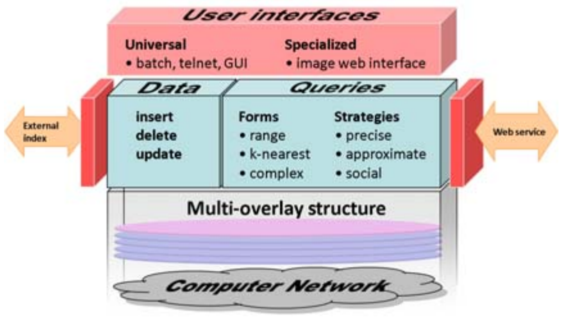

Fig. 3. Overview of MUFIN [2]

the system is running on. The executive core of MUFIN in the second tier is formed by several distributed indexing structures (*overlays*) that exploit the paradigm of P2P networks both in their structured and unstructured variants. Each of these overlays maintains data specific to it and distributes them among its (logical) peers. For example, an overlay can be defined for shape descriptors or color histograms in case of images, or protein spectra vectors in case of biological data. The number of logical peers in respective overlays and their mapping to physical computers are the main parameters that affect the system's searching performance.

From the third tier point of view, the logical peers of all overlays form a single virtual overlay with a uniform access to individual members. More precisely, the logical peers of different overlays mapped to the same physical host constitute a peer of this virtual overlay. The third tier provides interfaces for data maintenance (inserting and deleting data) and query specification, considering both the query form (range, nearest-neighbors or complex queries) as well as the strategy for query execution. Possible strategies are precise and approximate query evaluation. From the system point of view, these interfaces come in the form of a native API, a web service interface, or a plug-in interface for linking with external services.

Finally, the top-level tier represents interfaces allowing regular users to interact with the system. We have defined several general-purpose interfaces suitable for any application domain. However, they lack the comfort of a specialized interface. In Section 5.1, we present an example of an interface specific for image retrieval.

2.3 Properties

MUFIN is built by means of the Metric Similarity Search Implementation Framework (MESSIF) [4] – a large and extending Java library of metric searching implementation tools. It gives MUFIN flexibility in applying suitable implementation strategies for specific purposes and fast adoption of new progressive solutions as they come from research. The properties of MUFIN can be summarized by the following attributes:

Extensibility: Different similarity search indexes for specific applications can be built with a single tool;

Scalability: Due to the underlying P2P technology, extremely large datasets can be processed;

Adaptability: In highly-volatile or unreliable environments, self-organizing principles can be implemented and the system can operate in unstructured P2P networks;

Multi-modal Queries: In order to adjust effectiveness of search according to needs of individual users, several overlays can be combined together using a monotonic aggregation function [5];

Approximation: To further improve performance, approximation techniques can be applied to query evaluation [6,7];

Infrastructure Independence: The networking module uses standard IP protocols. Each peer is identified only by its IP address and port number, so the mapping of the system to a hardware infrastructure is extremely flexible. For example, an instance of MUFIN can operate on a local network of common workstations, on a single multiprocessor machine, on a world-wide network, or even on a GRID system.

3 Structured Networks

In this section, we focus on indexing mechanisms of MUFIN that create purely-decentralized and structured P2P networks. In general, each peer of such a system consists of the following components and expects them from the other peers: (i) resources – storage and computational power, (ii) communication – a peer can contact any other peer directly if it knows its network identification, and (iii) navigation – internal structure that ensures correct routing among the peers. To ensure maximum scalability, the system also adopts requirements of the Scalable and Distributed Data Structures [8]: (i) data expands to new peers gracefully if and only if the peers already used are efficiently loaded, (ii) there is no master site to be accessed when searching for objects, e.g., there is no centralized directory, and (iii) the data access and maintenance primitives (search, insert, split, etc.) never require atomic updates to multiple peers.

In the following, we describe a space-partitioning technique called Generalized Hyperplane Tree Star and a space-transformation technique named Metric Chord.

3.1 GHT*

The Generalized Hyperplane Tree Star (GHT*) [9] is a decentralized structured P2P network that distributes data to peers based on the generalized hyperplane partitioning principle. Each peer maintains a tree structure called *Address Search Tree* (AST). An example of AST with the corresponding space partitioning is

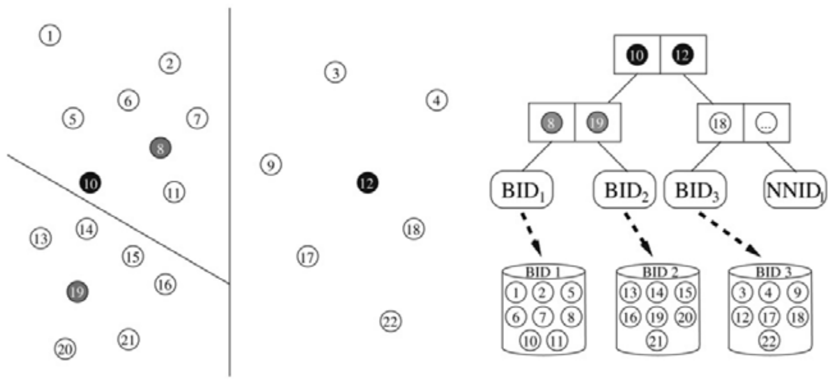

Fig. 4. Address Search Tree with the generalized hyperplane partitioning [9]

depicted in Figure 4. Internal nodes of AST store routing information – the definition of hyperplane. In metric spaces, the generalized hyperplane is defined using two objects p_1, p_2, so-called pivots. The data objects $o \in \mathcal{M}$ such that $d(o, p_1) \leq d(o, p_2)$ form the left partition whereas the other objects form the right partition. Leaf nodes of AST store pointers to local buckets (denoted as BID) or to other peers (denoted as NNID). A bucket is a limited storage space dedicated for data objects, e.g., a memory segment or a disk block. The number of buckets managed by a peer depends on its own potential and capacity. Since the structure is dynamic and new objects can be inserted at any time, a bucket on a peer may reach its capacity limit. In this situation, a new bucket is created and objects are redistributed between these two buckets following the hyperplane newly defined. The new bucket may also be allocated on a different peer. Thus, the structure grows as new data come in.

The core of the algorithm lays down a mechanism for locating the respective peers that hold requested objects. Whenever a peer wants to query or modify the data, it must first consult its own AST to get locations, i.e. peers, where the data resides. Then, it contacts the peers via network communication to actually process the operation. Since we are in a distributed environment, it is practically impossible to maintain a precise address for every object in every peer. Thus, the ASTs at the peers contain only limited navigation information which may be imprecise. The locating step is repeated on the contacted peers whenever AST is imprecise until the desired peers are reached. The algorithm guarantees that the destination peers are always found. The structure provides a mechanism called image adjustment for updating imprecise parts of AST automatically.

3.2 M-Chord

The Metric Chord (M-Chord) [10] is a decentralized structured P2P network as well but it applies a space transformation rather than a space partitioning. The

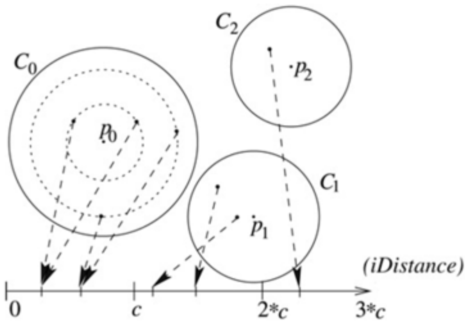

Fig. 5. The mapping principle of M-Chord [10]

transformation maps original objects to numeric identifications that are consequently organized in B$^+$-tree. In particular, a set of objects $p_0, \ldots, p_{n-1}$ (pivots) are selected and the following transformation based on distances is defined:

$$idistance(o) = d(o, p_i) + i \cdot c.$$

The distance of object o to the closest pivot p_i is determined and along with the separation constant c the numeric address is obtained. Figure 5 visualizes this mapping.

Having the data space mapped into the one-dimensional domain, each peer of the system takes over responsibility for an interval of keys. The structure of the system is formed by the Chord circle [11]. This P2P protocol provides an efficient localization of the peer responsible for a given key. When inserting a new object into the structure, the initiating peer computes the idistance value and employs Chord to forward a store request to the peer responsible for the corresponding interval. The peers store data in B$^+$-tree. When a peer reaches its storage capacity limit, it requests a split. A new peer is placed on the Chord circle, so that the requester's storage is split evenly.

3.3 Scalability Evaluation

In this section, we summarize experience with the approaches described above. We focus mainly on the scalability issue and concurrent query processing. A complete comparison made from other perspectives is available in [9].

Both the structures were implemented as overlays in MUFIN, which allows us to compare them objectively. We used a real-life dataset consisting of 1 million images taken from the CoPhIR dataset [12], for details please refer to Section 5.1. In M-Chord, the transformation using 40 pivots was defined and the capacity of peers' storage was fixed to 5,000 objects. In case of GHT*, the peers could maintain up to five buckets each of capacity of 1,000 objects. All presented performance characteristics of query processing have been obtained as an average over 100 queries with randomly chosen query objects and the radii of 0.8 (about 100 objects returned).

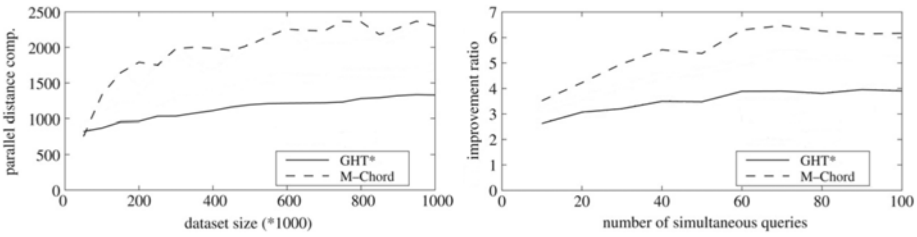

Fig. 6. Increasing data volume: (left) costs in parallel distance computation costs, and (right) query-throughput improvement ratio

Figure 6(left) presents the computational costs in terms of the number of parallel distance computations, i.e. it corresponds to the query response time. The costs grow very slowly. This is caused by the following facts: the peers involved in searching contain more data; and the data space got denser when the volume of data was increased. The noticeable graph fluctuations are caused by quite regular splits of overloaded peers. Figure 6(right) depicts the query-throughput improvement ratio that measures how many queries can be evaluated concurrently without degradation of response time. The differences in the respective improvement ratios are introduced mainly by differences between single-query parallel costs of individual structures. M-Chord handles simultaneous queries noticeably better than GHT*. GHT* employs quite a high number of peers during the query processing, so parallel distance computations are low (see Figure 6(left)). Therefore, simultaneous queries hit the same peers very likely, which increases the overall response time. Furthermore, there is a higher probability in M-Chord that different queries incur load at different peers and, thus, the parallel costs are only marginally increased.

MUFIN inherently supports also centralized index structures. So, the performance of distributed structures can be further improved by organizing peers' local data in a centralized index structure. For example, a very popular solution is to apply M-tree [13] or D-index [14].

4 Unstructured Networks

A technology based on Semantic Overlay Networks (SONs) [15,16,17], which creates a semantic overlay upon an existing unstructured network (e.g. Gnutella), has proven to be useful. The peers sharing similar interests are grouped into semantically similar clusters to improve query performance, while keeping a high degree of peer autonomy. An emerging research direction is to apply principles of self-organizing systems originating from different disciplines such as biology or social sciences. In general, self-organizing systems are characterized by a high degree of scalability, adaptability to changing environment, and robustness to sudden errors. Existing approaches [18,19] applied to search in unstructured networks usually adopt a self-organizing theory of biological systems – the ant-colony system or the social-network theory.

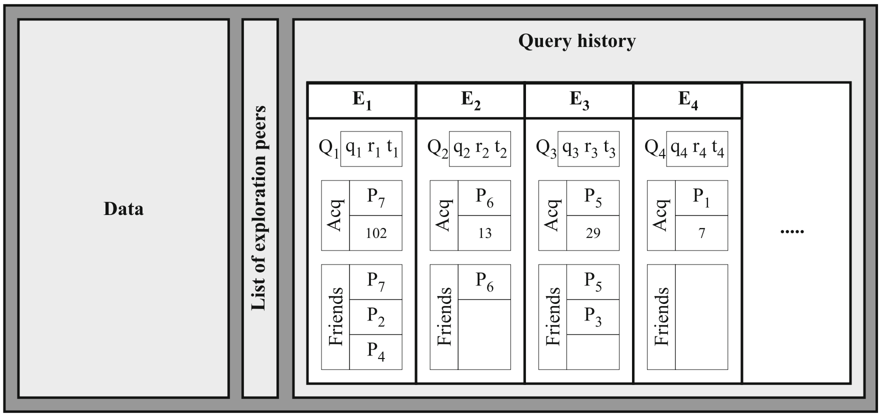

Fig. 7. MSN peer's schema

In this section, we outline a search system called Metric Social Network (MSN) [19]. In particular, MSN can be observed as an overlay implemented as a MUFIN's overlay that operates as an unstructured P2P network. MSN exploits the social-network paradigm [20,21] to lay basics for self-organizing principles – the relationships among peers are established according to analyses of answers of processed queries. An adaptive routing algorithm exploits these relationships for efficient query forwarding. The major difference from the structured-network approaches is that no data distribution principle is imposed, so data need not be transferred to another peer for storage.

Firstly, we summarize the MSN's architecture. Next, we describe its routing algorithm. Finally, we present a sketch of performance evaluation.

4.1 Architecture

Each peer of MSN can organize its own data, can pose similarity queries and must return answers to the queries. Interconnection between peers is based on the query-answer paradigm, i.e., new relationships among peers are established according to answers returned to a processed query. Thus, each peer maintains metadata about queries it has asked or answered, called a *query history*. This represents peer's local knowledge about the network and is exploited by a query-routing algorithm.

A peer P is a tuple (X, H, M), where X identifies the peer's local database, and $H = \{E_1, \ldots, E_n\}$ represents the query history. Individual *entries* E_i identify peers that participated in answering a query Q and form query-specific relationships. In addition, each peer maintains a list of peers M that are employed to explore new and previously unvisited parts of the network. The schema of a peer is depicted in Figure 7.

When a query Q is issued at a peer P_{start}, the routing algorithm tries to locate the most *promising* peers $P_1, \ldots, P_n$ in the network. These peers process the query on their local data and return their answers (*partial answers*) $A_{P_i}(Q)$ to the peer P_{start}. This peer merges the partial answers and returns the *combined answer* to the user, denoted as $A(Q) = \bigcup_{i=1}^{n} A_{P_i}(Q)$. Remark that the combined answer is approximate. To determine which peer answered better, the quality of the partial answers has to be measured. Even though sophisticated quality measures can be defined, MSN uses the quality of peer's answer expressed simply as the number of retrieved objects, i.e. $|A_{P_i}(Q)|$.

Two kinds of relationships are distinguished. Firstly, the acquaintanceship denotes that the target of the relationship is the best peer (*acquaintance*) to answer the given query. The acquaintance has the highest quality of the answer to the query Q and is defined as follows:

$$Acq(Q) = P \Leftrightarrow \forall P_i : |A_P(Q)| \geq |A_{P_i}(Q)|,$$

for $i \in \{1, \ldots, n\}$ where n denotes the number of peers answering the query Q. Secondly, the friendship represents the similarity of peers – two peers are *friends* when they give a similar (high-quality) answer to the query Q.

$$Fri(Q) = \{P_i \;:\; |A_{P_i}(Q)| \geq |A(Q)|/n\}.$$

Note that the acquaintance and the best friend are the identical peer.

After processing the query Q, each peer P_i identified as a friend stores a new entry E in its query history. This entry $E = (Q, Acq(Q), |A_{Acq(Q)}(Q)|, Fri(Q))$ is a tuple, where $Q = R(q, r, t)$ denotes the range query with timestamp, $Acq(Q)$ is the acquaintance, $|A_{Acq(Q)}(Q)|$ is its quality, and $Fri(Q)$ is the set of friends. The query-issuing peer P_{start} and peers contacted as exploration peers store this entry as well, but the set of friends is empty unless the particular peer has also been identified as a friend.

4.2 Adaptive Query Routing

In this part, we describe an adaptive query-routing algorithm proposed in [22] that enables each peer to control routing according to its current knowledge. In principle, each peer that is asked to process a query checks its query history for the most relevant entries. Next, the peer forwards the query to the acquaintances of these entries or evaluates the query on the local data and contacts friends. If there are few relevant entries only or there are not any, the routing algorithm uses exploration peers to locate unvisited peers that may contain the required data.

Relevancy of Entries. The relevancy of entries is measured by *confusability* of two queries – the query being evaluated and a query stored within an entry in the query history. The confusability function is a continuous function and returns a real value within $[0, 1]$. The higher the value is returned, the more confusable (relevant) the queries are. If it returns 1, the queries are identical. The function takes into account the distance between query objects of queries, their query radii and the time when the queries were issued. The time aspect is important to allow aging information about the peer's neighborhood. The formal definition called adaptive gaussian-like confusability is available in [22].

Exploration. The design of MSN incorporates factors to improve quality of query answers and to allow new peers to join the system efficiently. Each peer of MSN maintains its list of exploration peers over time [23]. At the beginning, it has to know at least one existing peer in order to be able to forward a query to other peers. The routing algorithm exploits this list in a way that it contacts not only the most promising peers retrieved from the query history but also some exploration peers that help find new and unvisited parts of the network.

Routing Algorithm. In general, a new query is being forwarded to the peers that should have better *knowledge* about the query – knowing more-promising peers or containing relevant data. The peer's knowledge is interpreted as confusability (P^{conf}) and for query-issuing peer P_{start} is set to zero ($P^{conf}_{start} = 0$). Firstly, P_{start} goes through its query history, computes the values of confusability between a new query $Q = R(q, r, t)$ and queries of all stored entries, and returns the entries descendingly ordered by confusability. Secondly, the list of relevant entries E_{rel} is constructed. All entries having confusability ≥ 0.8 are added to E_{rel}, because they are highly relevant to Q. If there are fewer entries in E_{rel} than 5, next entries having confusability ≥ 0.3 are added to fill up E_{rel} to contain five entries. Next, each entry in E_{rel} is processed as follows:

– If the entry has confusability C higher than the current peer's confusability P^{conf}, the query is forwarded to the acquaintance P_{acq} picked from this entry and its confusability P^{conf}_{acq} is set to C.
– Otherwise the query is not forwarded and is processed on local data. In addition, friends of entries in E_{rel} that have confusability ≥ 0.8, are asked to evaluate Q on their local data too. It is supposed that these friends hold substantial parts of the total answer $A(Q)$. The partial answers are finally returned to P_{start}.

If the list E_{rel} is shorter than five entries or even does not contain any entry, Q is forwarded to up to five exploration peers. To avoid flooding the network, forwarding to exploration peers is stopped after a predefined number of hops is reached (in our case, 3 hops). The complete specification of adaptive query routing algorithm is available in [22].

4.3 Adaptability and Robustness Evaluation

In order to study characteristics of the query routing algorithm, we have implemented MSN in MUFIN and executed real-life experiments. We used 100,000 images taken from the CoPhIR dataset [12], described in details in Section 5.1. Each of image has its owner ID associated, so we distributed images over P2P network in a way that each peer contains images of one Flickr user. Because there are high differences in the number of images taken by individual users, we have split overfilled peers. As a result, we obtained 2,000 peers each organizing 50 images of the same Flickr user.

Figure 8(left) reports on the results obtained by repeating 100 times a batch of queries and measuring performance indicators. In particular, we measured

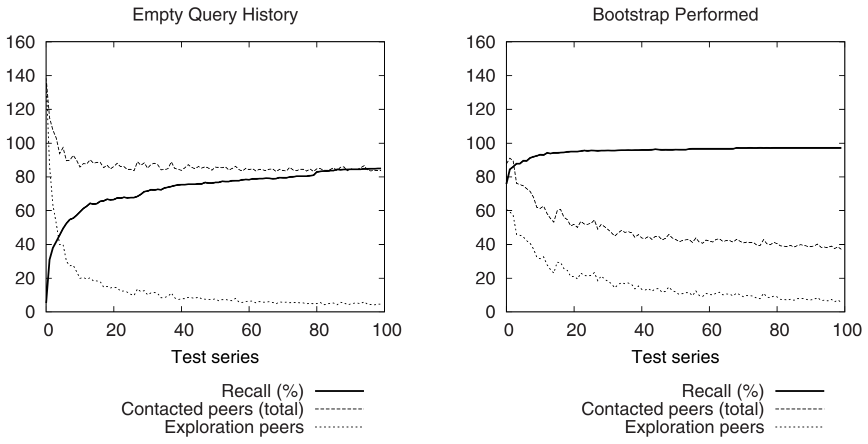

Fig. 8. Performance indicators of MSN: (left) query history is initially empty, and (right) query history is populated with 3 random queries

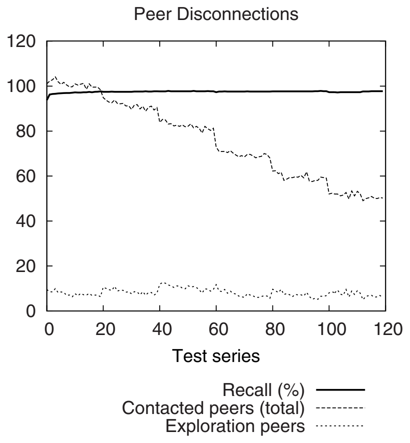

Fig. 9. Performance indicators of MSN: gradual disconnection of peers

recall, the number of exploration peers used during querying, and costs in terms of contacted peers (peers participated in query evaluation including exploration peers). The batch consisted of 50 range queries with randomly picked query objects and varying radii. From the figure, we can read that the system starting from zero knowledge (peers had query histories empty) started to evolve and the recall has reached 85% while contacting less than 90 peers. Initially, each peer had just 50 exploration peers, so the peers could use only exploration peers for query routing.

Performance of MSN can radically change if a peer joining the system proceeds a bootstrap procedure. Figure 8(right) shows the same experiment but the peers performed the following bootstrap procedure during their joining. Firstly, three objects were picked at random from the peer's local data. Secondly, range queries with these objects and radius 0.8 were posed. Finally, the MSN evaluated the queries. This helps distribute the knowledge about new peer's local data in the network. As a result, the first batch execution reached 80% recall. After 100^{th} batch execution, the recall was 97% and the costs decreased to 37 contacted peers. In this way, the quality of service of MSN is greatly improved.

We have also tested robustness of MSN by gradually disconnecting up to 1,000 peers. Figure 9 shows the same performance indicators when 200 random peers got disconnected forcibly after each 20^{th} batch execution. The most interesting fact about the recall curve is that it stays almost constant. This proves adaptability of the query routing algorithm and robustness of the whole system. The answer was degrading in terms of amount of retrieved data, but only because some data became unavailable.

5 Prototype Applications

As mentioned in the previous sections, the similarity search approach used in MUFIN is highly universal and extensible. In this section, we describe several application domains where MUFIN can be used. However, due to MUFIN's versatility this list is not complete. Firstly, we present a large-scale image retrieval demo. Next, we summarize other applications and give ideas how to incorporate them into MUFIN.

5.1 Large-Scale Image Search

This application [24] represents a possible instance of MUFIN for content-based similarity search in a large collection of general images available on the internet. In particular, the dataset consists of 100 million images taken from CoPhIR Database [12]. Each image is represented by five global MPEG-7 descriptors [25], namely *color structure* (CS), *color layout* (CL), *scalable color* (SC), *edge histogram* (EH), and *homogeneous texture* (HT). Specifically, CS, CL, and SC express the spatial distribution of colors in an image. The EH captures local density of edge elements and their directions (sometimes called the *structure* or *layout*); it acts as a simple and robust representation of shapes. Finally, HT is a texture descriptor. These descriptors are represented as vectors and the MPEG-7 standard defined a specific distance measure for each of them. These measures satisfy the metric postulates and they are aggregated into a single distance function. The whole dataset is organized in M-Chord and peers' local data are stored in M-tree. For details, please refer to Section 3.2. An example of retrieving k images which are the most similar to a given query image is given in Figure 10. For further details, please refer to [2,7].

5.2 Biometric Applications

In general, biometrics are automated methods of recognizing a person based on the person's physiological and behavioral characteristics. Biometrics include a

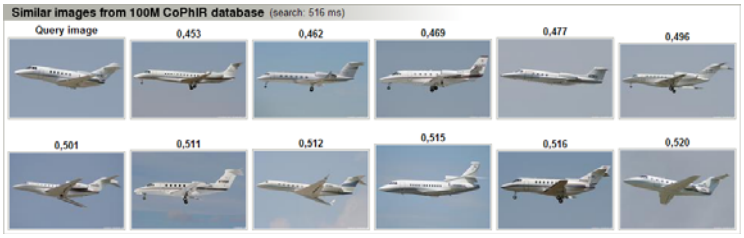

Fig. 10. Image Retrieval: the result of a query

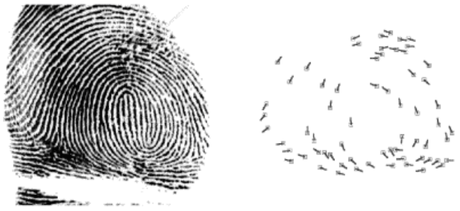

Fig. 11. Example of minutiae extracted from a fingerprint image [26]

wide variety of technologies ranging from traditional fingerprints over facial or iris recognition and retinal scanning to DNA testing, speech verification and gait recognition. MUFIN can be applied to the problem of *identification*, the aim of which is to tell who the person that exposes its biometric characteristic is.

A famous application of biometrics is in criminalistics and in border and immigration control where fingerprints are compared. Minutiae is one of the successfully applied methods of comparing ridges in fingerprints [26]. It identifies places where ridges start, stop or bifurcate (branch), refer to Figure 11. These places are then observed as points with a direction and are converted to polar coordinates. As a result, a fingerprint is described as a sequence of points. Two sequences are then matched using a weighted edit distance function. The used weights do not break metric postulates, so this distance function is directly applicable to MUFIN.

Gait, or the way a person walks, is a unique and idiosyncratic characteristic of the person. Its advantage for biometrics is that it is difficult to conceal and it can be easily captured even at long distances. In [27], the gait information is extracted from a video sequence. In particular, a silhouette of the walking person is determined for each video frame by subtracting the background of the image. The sequence of silhouettes is divided in subsequences each of them representing one gait cycle (two steps). Then, an average silhouette is computed for each subsequence, see Figure 12. The binary silhouettes are then compared using the Euclidean distance, which is metric.

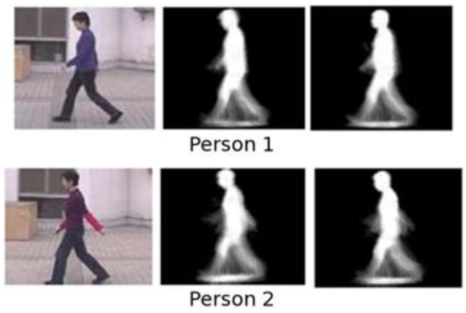

Fig. 12. Example of average silhouette extraction [27]

6 Conclusions

There are no doubts that modern similarity search in computer networks needs new technology to apply. In this paper, we have shortly introduced MUFIN, an approach to similarity searching, which is designed on concepts of: (i) extensibility - to achieve applicability to different collections comparing data by various measures of similarity; (ii) scalability - to process extremely large collections of data queried by many concurrent requests; (iii) infrastructure independence - to tune performance according to needs of specific applications. The implementation on structured P2P networks is able to achieve quality of service by tuning the performance according to specific application needs. We discuss several structured P2P protocols, all of them running with logarithmically bound number of hops. Local data on peers is organized in centralized metric similarity search structures. Unstructured P2P networks with high degree of peer churning are considered as systems of self-organizing peers for which a social network of search requests and answers is built. Such architecture can learn and improve its effectiveness in time; it is also able to react to the changing number of peers properly. Important features are demonstrated by an on-line demo available from `http://mufin.fi.muni.cz/imgsearch/`.

Acknowledgments. This research was partially supported by the Czech Science Foundation projects 201/09/0683 and 201/07/P240. The access to the MetaCentrum (super)computing facilities provided under the research intent MSM6383917201 is also appreciated.

References

1. Novak, D., Batko, M., Zezula, P.: Generic similarity search engine demonstrated by an image retrieval application. In: The 32nd Annual International ACM Conference on Research and Development in Information Retrieval, p. 840. ACM Press, New York (2009)

2. Batko, M., Dohnal, V., Novak, D., Sedmidubsky, J.: MUFIN: A Multi-Feature Indexing Network. In: The 2nd International Workshop on Similarity Search and Applications, pp. 158–159. IEEE Computer Society, Los Alamitos (2009)
3. Zezula, P., Amato, G., Dohnal, V., Batko, M.: Similarity Search: The Metric Space Approach. In: Advances in Database Systems, vol. 32. Springer, Heidelberg (2006)
4. Batko, M., Novak, D., Zezula, P.: MESSIF: Metric similarity search implementation framework. In: DELOS Conference 2007: Working Notes, pp. 11–23. Information Society Technologies (2007)
5. Batko, M., Kohoutková, P., Zezula, P.: Combining metric features in large collections. In: The 1st International Workshop on Similarity Search and Applications, pp. 79–86. IEEE Computer Society, Los Alamitos (2008)
6. Amato, G., Rabitti, F., Savino, P., Zezula, P.: Region proximity in metric spaces and its use for approximate similarity search. ACM Transactions on Information Systems 21(2), 192–227 (2003)
7. Novak, D., Batko, M., Zezula, P.: Web-scale system for image similarity search: When the dreams are coming true. In: The 6th International Workshop on Content-Based Multimedia Indexing, pp. 446–453. IEEE, Los Alamitos (2008)
8. Litwin, W., Neimat, M.A., Schneider, D.A.: LH* – a scalable, distributed data structure. ACM TODS 21(4), 480–525 (1996)
9. Batko, M., Novak, D., Falchi, F., Zezula, P.: Scalability comparison of peer-to-peer similarity search structures. Future Generation Computer Systems 24(8), 834–848 (2008)
10. Novak, D., Zezula, P.: M-Chord: A scalable distributed similarity search structure. In: The 1st International Conference on Scalable Information Systems, pp. 1–10. IEEE Computer Society, Los Alamitos (2006)
11. Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for internet applications. In: The 2001 ACM Conference on Applications, Technologies, Architectures, Protocols for Computer Communications, pp. 149–160. ACM Press, New York (2001)
12. Bolettieri, P., Esuli, A., Falchi, F., Lucchese, C., Perego, R., Piccioli, T., Rabitti, F.: CoPhIR: a test collection for content-based image retrieval. CoRR, abs/0905.4627v2 (2009)
13. Ciaccia, P., Patella, M., Zezula, P.: M-tree: An efficient access method for similarity search in metric spaces. In: The 23rd International Conference on Very Large Data Bases, pp. 426–435. Morgan Kaufmann, San Francisco (1997)
14. Dohnal, V., Gennaro, C., Savino, P., Zezula, P.: D-Index: Distance searching index for metric data sets. Multimedia Tools and Applications 21(1), 9–33 (2003)
15. Aberer, K., Cudré-Mauroux, P.: Semantic overlay networks. In: The 31st International Conference on Very Large Data Bases, p. 1367. ACM Press, New York (2005)
16. Bender, M., Crecelius, T., Kacimi, M., Michel, S., Parreira, J.X., Weikum, G.: Peer-to-peer information search: Semantic, social, or spiritual? IEEE Data Eng. Bull. 30(2), 51–60 (2007)
17. Crespo, A., Garcia-Molina, H.: Semantic overlay networks for p2p systems. In: Moro, G., Bergamaschi, S., Aberer, K. (eds.) AP2PC 2004. LNCS (LNAI), vol. 3601, pp. 1–13. Springer, Heidelberg (2005)
18. Michlmayr, E.: Self-organization for search in peer-to-peer networks: the exploitation-exploration dilemma. In: The 1st international conference on Bio inspired models of network, information and computing systems, p. 29. ACM Press, New York (2006)

19. Sedmidubsky, J., Bartoň, S., Dohnal, V., Zezula, P.: A self-organized system for content-based search in multimedia. In: The IEEE International Symposium on Multimedia, pp. 322–327. IEEE Computer Society, Los Alamitos (2008)
20. Wasserman, S., Faust, K., Iacobucci, D.: Social Network Analysis: Methods and Applications (Structural Analysis in the Social Sciences). Cambridge University Press, Cambridge (1994)
21. Granovetter, M.: The strength of week ties. American Journal of Sociology 78(6), 1360–1380 (1973)
22. Dohnal, V., Sedmidubsky, J.: Query routing mechanisms in self-organizing search systems. In: The 2nd International Workshop on Similarity Search and Applications, pp. 132–139. IEEE Computer Society, Los Alamitos (2009)
23. Sedmidubsky, J., Bartoň, S., Dohnal, V., Zezula, P.: Querying similarity in metric social networks. In: Enokido, T., Barolli, L., Takizawa, M. (eds.) NBiS 2007. LNCS, vol. 4658, pp. 278–287. Springer, Heidelberg (2007)
24. Batko, M., Falchi, F., Lucchese, C., Novak, D., Perego, R., Rabitti, F., Sedmidubsky, J., Zezula, P.: Building a Web-scale Image Similarity Search System. Multimedia Tools and Applications, 31 (2009)
25. Manjunath, B.S., Salembier, P., Sikora, T. (eds.): Introduction to MPEG-7: Multimedia Content Description Interface. John Wiley & Sons, Inc., New York (2002)
26. Jain, A.K., Maltoni, D.: Handbook of Fingerprint Recognition. Springer-Verlag New York, Inc., Secaucus (2003)
27. Fazenda, J., Santos, D., Correia, P.: Using gait to recognize people. In: The International Conference on Computer as a Tool, vol. 1, pp. 155–158. IEEE Press, Los Alamitos (2005)