SIXTH FRAMEWORK PROGRAMME
PRIORITY 2
"Information Society Technologies"

Deliverable D5.4

Executing complex similarity queries over multi layer P2P search structures

March 2009

Project acronym: SAPIR
Project full title: Search on Audio-visual content using P2P Information Retrieval
Contract no.: 45128

Deliverable type: Prototype
Classification: Public
Work package and task: WP5, T5.4
Responsibility: ISTI-CNR

Editor: Fabrizio Falchi (ISTI-CNR)

Contributors: Michal Batko (Mu-Brno), Fabrizio Falchi (ISTI-CNR)

Internal Reviewer: Mouna Kacimi (MPI)

EXECUTIVE SUMMARY

This deliverable reports the activities conducted within Task 5.4 *“Executing complex similarity queries over multi layer P2P search structures”* of the SAPIR project. In particular the deliverable discusses complex similarity queries issues and the implementation of the query processing over the P2P indexing. The document is accompanied by a zip file containing the javadoc for MUFIN.

TABLE OF CONTENTS

1 INTRODUCTION

The aim of the SAPIR project is to develop novel and scalable methods for querying audio-visual content using complex queries combining text, meta-data and audio-visual content. Query input is supplied by the user following the “query by example” paradigm using several audio-visual sources for querying. Scalability is achieved by using P2P overlay network with Semantic Overlay Networks (SON) for the different media involved. Example Overlays can be an image overlay that index and search images in a P2P overlay using metric space for Visual descriptors similarity. Another overlay can deal with text and metadata.

1.1. OBJECTIVES OF WP5

The aim of WP5 is to develop novel and scalable methods for content-search in audio-visual data. Query input is supplied by the user following the “query by example” paradigm using several audio-visual sources for querying. P2P overlay networks are used to provide scalable search and combine sophisticated ranking algorithms over several media specific similarity metrics.

1.2. OBJECTIVES OF THE DELIVERABLE

The objective of this deliverable is to report the activities conducted during Task 5.4 - *“Executing complex similarity queries over multi layer P2P search structures”*. The main objective of the Task was to achieve multi-feature similarity ranking based on the P2P similarity indices developed in WP4. In this report we describe the implementation of scalable algorithms to answer top-k queries for multiple predicates supported by distributed similarity search index overlay developed in T4.1, on a dynamic network of peer computers.

The deliverable is organized as follows. In Section 2 we give an overview of the similarity search problem focusing on complex queries and on a general distributed incremental nearest neighbor algorithm. In Section 3 we describe the MUFIN overlay and its implementation. Moreover we describe the distributed threshold algorithm specifically developed for MUFIN. In Section 4 how the metadata of the CoPhIR collection was used for indexing using MUFIN. This document is accompanied by a zip file containing a javadoc for MUFIN.

2 SIMILARITY SEARCH

2.1 INTRODUCTION

Similarity search, particularly in metric spaces, has been receiving increasing attention in the last decade, due to its many applications in widely different areas. The notion of similarity has been studied extensively in the field of psychology and the given definition characterizes that similarity has been found to have an important role in cognitive sciences. From a database prospective, similarity search is based on gradual rather than exact relevance. A distance between objects is used to quantify the proximity, similarity or dissimilarity of a query object versus the objects stored in a database.

A similarity search can be seen as a process of retrieving data objects in order of their distance or dissimilarity from a given query object. It is a kind of sorting, ordering, or ranking of objects with respect to the query object, where the ranking criterion is the distance measure. The most common similarity queries are Range and Nearest Neighbor (see D4.1 - Section 2.1). Though this principle works for any distance measure similarly to most recent research in this topic, we restrict the possible set of measures to metric distances (see D4.1, Section 2.2). In fact, because of the mathematical foundations of the metric space notion, partitioning and pruning rules can be defined in this case for developing efficient index structures. In Deliverable D4.1 we discussed the similarity search concept in more detail.

2.2 COMPLEX SIMILARITY QUERIES

Complex similarity queries are queries consisting of more than one similarity predicate. Efficient processing of such kind of queries differs substantially from traditional (Boolean) query processing. The problem was studied first by Fagin in [Fagin 1996]. The similarity score (or grade) a retrieved object receives as a whole depends not only on the scores it gets for individual predicates, but also on how such score are combined.

To this aim, Fagin has proposed two algorithms - the A0 algorithm [Fagin 1996] and the Threshold algorithm [Fagin 2001]. Both algorithms assume that for each query predicate we have an index structure able to return objects in order of decreasing similarity. This is an important example of the application of Incremental Nearest Neighbor.

By using the Fagin algorithms it is possible to answer queries involving more than one feature overlay, such as: find all images most similar to the query image with respect to the color and the shape at once. In this situation, we do not know how many neighbors must be retrieved in individual layers before the best object is found that satisfies the complex condition.

In Section 2.3 we describe a generic Distributed Incremental Nearest Neighbor (DINN) algorithm which approach the Incremental Nearest Neighbor problem for P2P-based systems. The DINN algorithm finds closest objects in an incremental fashion over data distributed among computer nodes, where each node is able to perform its local Incremental Nearest Neighbor (local-INN) algorithm. The algorithm is optimum with respect to both the number of involved nodes and the number of local-INN invocations.

In Section 3.3 we present a distributed threshold algorithm specifically developed for MUFIN. It allows a user to specify an arbitrary aggregation function for combining the individual descriptor distances into an overall distance so that the best matching results are closest to the query. The function can be different for each query thus different results can be obtained even if the query descriptors remain the same.

Another important contribution to the complex similarity queries is the work presented in [Ciaccia et al. 1998]. In this paper the authors concentrated on complex similarity queries expressed through a generic language and whose predicates refer to a single feature. The proposed solution suggests that the index should process complex queries as a whole. The approach was implemented through an extension of the M-tree. Under the limitation of single feature, experimental results show that performance of the extended M-tree is consistently better than the A0 Algorithm. In the following we will concentrate on the multi feature problem.

2.3 DISTRIBUTED INCREMENTAL NEAREST NEIGHBOR ALGORITHM

An important building block of complex similarity queries algorithms, as the ones proposed by Fagin, is incremental similarity search. An incremental similarity search can provide objects in order of decreasing similarity without explicitly specifying the number of nearest neighbors in advance. As reported in Deliverable D4.1, when finding k nearest neighbors to the query object using a *kNN* algorithm, *k* is known prior to the invocation of the algorithm. Thus, if the *(k+1)*-th neighbor is needed, the kNN needs to be re-invoked for *(k+1)* neighbors from scratch. To resolve this problem, the authors of the Incremental Nearest Neighbors algorithm [Hjaltason and Samet 1999] proposed the concept of distance browsing which is to obtain the neighbors incrementally (i.e. one by one) as they are needed. This operation means browsing through the database on the basis of distance.

For allowing efficient execution of complex queries in distributed systems the definition of a generic Distributed Incremental Nearest Neighbor (DINN) was investigated [Falchi et al 2009] [Falchi et al 2007] during the SAPIR project. The proposed algorithm finds closest objects in an incremental fashion over data distributed among computer nodes, each able to perform its local Incremental Nearest Neighbor (local-INN) algorithm.

The DINN algorithm is based on a generalization of the algorithm proposed in [Hjaltason and Samet 1999]. The incremental nearest neighbor algorithm defined in [Hjaltason and Samet 1999] is applicable whenever the search space is structured in a hierarchical manner. The algorithm starts off by initializing the queue of pending requests with the root of the search structure — since the order of entries in this queue is crucial, they refer to it as the priority queue. Our proposed solution is independent of any specific P2P architecture – it can be applied to any Scalable and Distributed Data Structure (SDDS), P2P system, and Grid-based similarity search infrastructure.

In the main loop, the element closest to the query is taken off the queue. If it is an object, it reports it as the next nearest object. Otherwise, the child elements of the element in the search hierarchy are inserted into the priority queue.

The INN algorithm [Hjaltason and Samet 1999]] was defined for a large class of centralized hierarchical spatial data structures. Instead our DINN algorithm is distributed and not limited to hierarchical structures. Thus it can be used over SDDSs, P2P systems and Grid infrastructures. Our algorithm is built over nodes which are able to perform locally an INN between the objects they store (this will be formalized in Assumption 1).

In particular, we reformulate the definition of priority queue given in [Hjaltason and Samet 1999] by considering as elements of the queue, objects and nodes (or peers). We prove that our algorithm is optimal, in terms of both number of involved nodes and local-INN invocations. The elements of the queue are ordered according to a key which is always associated with both objects and nodes.

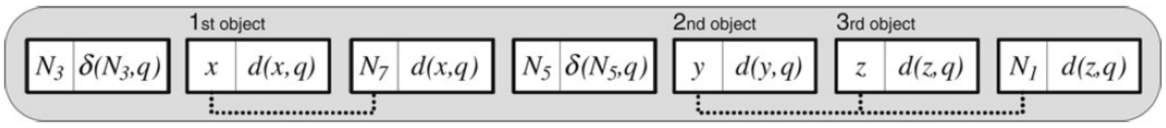


Figure 1 Snapshot of the priority queue during the execution of the DINN algorithm.

The key associated with each object is the distance between the query and the object itself. Instead the key associated with each node is a lower bound for the distance between the query and the next result coming from the node. While for an already involved node this lower bound can be simply the distance from the query of the last object retrieved by its local-INN, for the not yet involved nodes a naive solution could be to always use 0 as lower bound. However, this would imply all nodes to be involved for every similarity query. To avoid this, we suppose that each node is able to evaluate this lower bound for every node it knows (in P2P systems they are called neighbors).

Furthermore, in P2P systems there is no global knowledge of the network. Thus, we make an assumption (regarding the ability to find the next most promising node (by considering the lower bound mentioned before). This assumption replaces the consistency condition used in [Hjaltason and Samet 1999] for hierarchical data structures. We prove that our assumption can be satisfied under one of two simpler conditions which are common for data structures able to perform similarity search.

During the DINN algorithm execution, the queue contains a certain number of entries sorted in order of decreasing key. Entries can be both nodes and objects. Because of the values used as key, when a node is after an object we are sure that no better results than the object itself can be found in the node. The algorithm proceeds by processing the queue from the top. Basically if the first entry of the queue is an object, this object is the result of the DINN. In case the first entry is a node, we invoke its local-INN. The resulting object of this invocation is placed in the queue and its distance from the query allows us to update the entry with a more accurate (greater) lower bound which moves the node backward in the queue.

In [Falchi et al 2009, Falchi et al 2007] we proved that the DINN algorithm is optimum with respect to both the number of involved nodes and the number of local-INN invocations.

This outlined implementation is intrinsically sequential, since a single step of the algorithm involves only the first element of the queue at a time. In the second part of the paper, we straightforwardly generalize the algorithm introducing parallelism by invoking the local-INN algorithm of more than one node simultaneously. The precise definition of the algorithms is provided in the next section. Examples are given to help understanding the algorithm.

3 THE MUFIN OVERLAY AND ITS IMPLEMENTAION

3.1 DESCRIPTION

The MUFIN indexing overlay merges two similarity search technologies:

- M-Chord: peer-to-peer structure for similarity search in metric spaces [Novak, Zezula, 2006],
- M-Tree: a dynamic disk-oriented local indexing structure for similarity search in metric spaces.

Every *peer* (unit of the distributed structure) of the M-Chord organizes its data locally in M-Tree. Let us describe separately the principles of M-Chord and M-Tree.

3.1.1 M-Chord Principles

The M-Chord structure is based on the following ideas:

- It maps the metric space into a one-dimensional domain exploiting a set of selected pivots.
- It employs standard P2P techniques, like Chord or Skip Graphs, to divide the one-dimensional routing domain among the peers and to provide the navigation within the systems.
- Using the properties of the data space mapping, M-Chord provides algorithms for efficient evaluation of the metric similarity queries.

The fundamental idea of M-Chord is to map the metric space into a single-dimensional domain. The mapping schema has been inspired by iDistance, which is a centralized technique for *kNN* queries in vector spaces. We have generalized its fundamental idea for metric spaces and further adjusted it to meet our needs.

The M-Chord mapping, schematically depicted in Figure 2 (a), works basically as follows: Several reference points are selected from the sample dataset – we call these objects pivots and are denoted as p_i. The data space is partitioned in a Voronoi-like manner into clusters (C_i). Following the iDistance idea, the one-dimensional mapping of the data objects is defined according to their distances from the cluster's pivot. Having a separation constant c, the M-Chord key for an object $x \in C_i$ is calculated as

$$\text{mchord}(x) = d(p_i, x) + i \cdot c$$

To evaluate a range query $R(q; r)$, the data space to be searched is specified by mchord domain intervals in clusters which intersect the query region – see an example in Figure 2 (b).

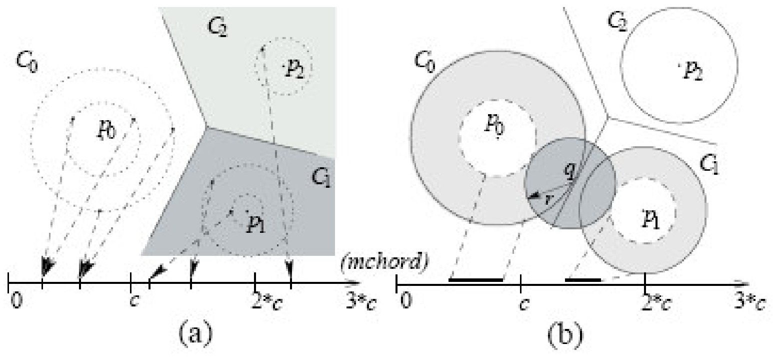


Figure 2: M-Chord mapping principles (a) and search-space pruning mechanism (b).

Having the data space mapped into the mchord domain, every active node (peer) of the system takes over responsibility for an interval of keys. The M-Chord then applies one-dimensional P2P protocol, for instance Chord or Skip Graphs, in order to divide the data among the peers and to provide navigation within the system. The M-Chord search algorithms navigate the similarity queries to the relevant peers. See Section 3.2 for details for detail on the M-Chord search algorithms.

3.1.2 The M-Tree Structure

The M-Tree [Ciaccia et al., 1997] is a popular dynamic disk-oriented structure for metric data indexing. Similarly to B-Trees and R-Trees, it is a balanced tree built in a bottom-up fashion by splitting overfilled nodes. Each entry of the M-Tree internal node contains a pivot and a covering radius which specify a sphere-like region of the space covering by the entry and its sub-tree. The leaf nodes store data objects together with their distances to the pivot in the parent node. The internal nodes keep distances to the parent node's pivot as well. All these values are utilized in order to achieve pruning effect for the search algorithms.

The M-Tree can evaluate standard similarity queries – the range query and the kNN query. The latter can be processed also in an approximate fashion. The approximate algorithm follows the generic scheme: It maintains a dynamic queue of M-Tree nodes sorted according to a heuristic which ensures that the "most promising" entries are processed first. If the currently processed entry is an internal node, its child nodes are reinserted into the queue; the leaf nodes are processed according to standard kNN algorithm. The processing can be stopped at any time according to a predefined condition; in our implementation, the approximate search is stopped when a certain portion of data has been searched.

A number of M-Tree extensions have been published [Zezula et al., 2006]. We implement the Pivoting M-Tree (PM-Tree) extension [Zezula et al., 2006], which employs additional filtering and pruning by means of pre-computed distances between the indexed objects and a fixed set of pivots. We use the same set of pivots as the M-Chord and thus the object-pivot distances are computed only once during the insert operation. The M-Tree is used as a local index at each peer of the M-Chord network. The overall system is schematically depicted in Figure 3.

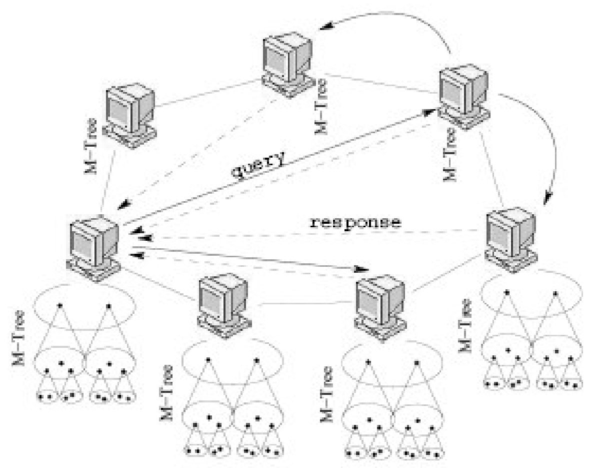


Figure 3: Schema of the MUFIN overlay.

3.2 IMPLEMENTATION

Let us now describe details of the M-Chord algorithms and of the implementation of the whole MUFIN overlay.

3.2.3 M-Chord Algorithms Details

Insert operation

Any node N_{ins} from the structure can initiate insert operation for an object *o*. First, node N_{ins} applies calculates *mchord*(*o*) key according to formula described above. Values $d(p_0,o),\ldots,d(p_{n-1},o)$ are obtained as a by-product and are carried along with *o* from now on.

The navigation protocol (e.g., Chord [Sotica, 2001], Skip Graphs [Aspnes, 2007]) is now followed to forward the insert request to node $N_{<mchord(o)>}$ responsible for key *mchord*(*o*). See an example of this process in Figure 4. This node stores *o* into a metric-based similarity index structure such as M-Tree.

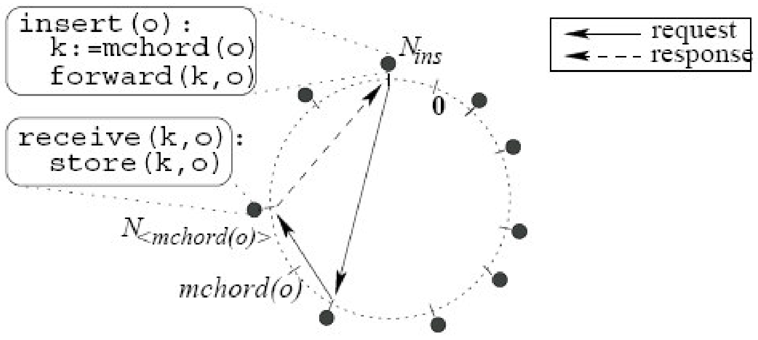


Figure 4: Principles of M-Chord Insert operation.

Range Query Operation

Resolving range queries is a basic task which is fundamental also for more complex similarity queries. The node N_q which initiates the query $R(q,r)$ runs the following RANGESEARCH(q,r) algorithm:

- for each cluster C_i, $i = 0,\ldots,n-1$, determine interval I_i of keys to be scanned:

  $I_i=[d(p_i,q) + i\cdot c-r),\ d(p_i,q)+ i\cdot c + r)]$;

- for $i= 0,\ldots,n-1$ use the P2P navigation algorithm to deliver the LOCALSEARCH(q,r) request to nodes whose intervals of keys intersect with I_i; two variants can be used:
  - o route the query to the node responsible for the “midpoint” of interval and then spread the message left and right (this variant is expected in the following),
  - o or branch the routing process to reach the target nodes at once;
- wait for all responses and create the final answer set.

LOCALSEARCH(q,r) is evaluated on the destination nodes as follows:

- search the locally stored data and create the local answer set

  $S_A = \{x \mid mchord(x) \in I_i,\ d(q,x) \le r\}$

- let the local metric-based index structure M-Tree evaluate the $R(q,r)$ query;
- send S_A back to node N_q.

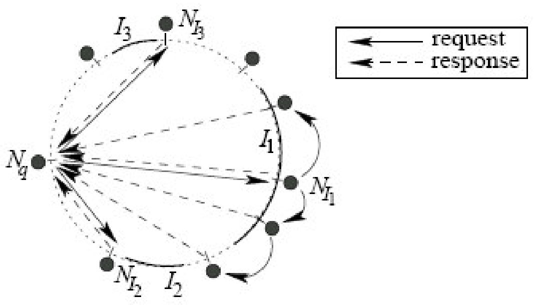


Figure 5: Principles of the M-Chord range search algorithm.

Figure 5: Principles of the M-Chord range search algorithm. shows an example of a range query flow through the system. The figure simplifies the query forwarding to nodes N_{I_i} as the routing is usually not direct. So, *n* queries are spread simultaneously from node N_q following the navigation protocol (Chord, Skip Graphs, etc.). According to this protocol, a lot of these messages "travel" partially along the same path. In order to decrease flooding of the network, the requests are sent as one message along the common path parts.

Approximate kNN Search

Precise processing of *kNN* queries can be relatively expensive. Therefore, M-Chord can evaluate *kNN* queries in an approximate fashion. The approximate algorithm is parametrized; the parameters can be set according to the particular dataset and the expected ratio between precision and efficiency.

Having *kNN* (*q,r*) query, the idea of the algorithm is to:

1. Determine a certain number *m* of pivots p_i which are the nearest to *q*. The corresponding *m* clusters are considered to be the most promising to contain objects near *q*.
2. For these pivots, calculate values $d(p_i,q)+i \cdot c$; these values correspond to areas where objects similar to *q* could occur.
3. The *kNN* (*q,k*) query is routed to peers responsible for the calculated values. These peers evaluate the query on their local index structures.
4. The query can also be spread to a certain number of the peers neighboring on either side of the navigation circle.

The number of nearest pivots involved in the searching, *m*, can be either fixed as a parameter or can be determined by a heuristic. The number of peers visited within each cluster can be a either fixed or defined as a "percentage of the peers which cover the cluster". The latter approach requires an external analysis of the particular instance of M-Chord to find out numbers of peers in individual clusters.

3.2.4 MESSIF: Implementation Framework

Implementation of all MUFIN parts (M-Chord, M-Tree, etc.) takes advantage of the Metric Similarity Search Implementation Framework (MESSIF) [Batko et al., 2007]. This Java-based framework allows a developer to focus on the implementation of the core of the technique and let the MESSIF modules take care of the low-level tasks like storage management, networking or performance statistics gathering. It pursues the following objectives:

- to provide *basic support* for the indexing based on metric space;

- to create a unified and semi-automated mechanism for *measuring* and collecting *statistics*;
- to define and use uniform interfaces to support *modularity* and thus allow *reusing of the code*;
- to provide infrastructure for *distributed processing* with focus on peer-to-peer paradigm – communication support, deployment, monitoring, testing, etc.;
- to support similarity search in *multi-metric* spaces (see the following section).

3.3 DISTRIBUTED THRESHOLD ALGORITHM FOR MUFIN

To be able to execute also combined search where a query is composed of more than one descriptor and its result is formed by objects that best match the query in several aspects, a distributed variant of the threshold algorithm has been proposed for MUFIN [Batko et al., 2008]. It allows a user to specify an arbitrary aggregation function for combining the individual descriptor distances into an overall distance so that the best matching results are closest to the query. The function can be different for each query thus different results can be obtained even if the query descriptors remain the same.

Assume that a complex data object is simplified by extraction techniques into a set of descriptors. Such a set together with an identifier of the original object form a metaobject.

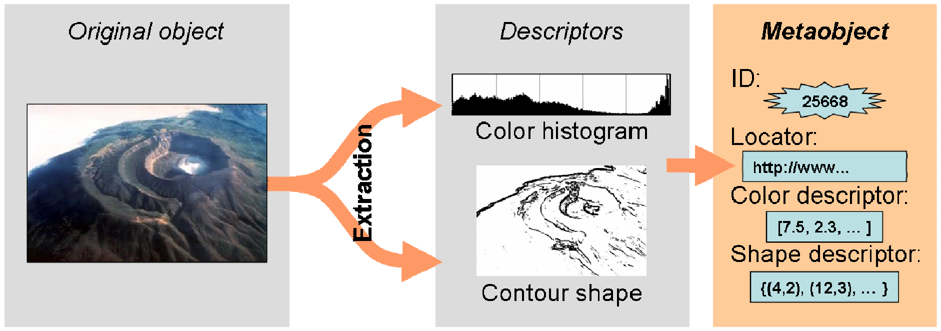


Figure 6: Example of metaobject extraction

To compare similarity within respective descriptors, a metric function is specified. For instance, we would like to index images from which we can extract color and shape descriptions. A metaobject for a given image is composed of the two descriptors – one for the color and the other for the shape – plus the link to the original image (see Figure 6). Naturally, to express the similarity of two metaobjects, we must combine the similarities of their respective descriptors. For this task, a user-supplied aggregation function f(x1, x2, ... xn) is used to compute the combined distance d(o1, o2) = f(d1(e1(o1), e1(o2)), d2(e2(o1), e2(o2)), ... , dn(en(o1), en(o2)) between two objects. The aggregation function is required to be monotonous. The result is not necessarily metric (consider for a example an aggregation function f(x1, x2, ... xn) = x1+…+xn +1. So d(o, o) = f(d1(e1(o), e1(o)), ... , dn(en(o), en(o))) = f(0, ..., 0) = 1 which contradicts the reflexivity property). Still it expresses the similarity in the same way as metric functions – the higher the value of distance the more dissimilar the objects are. Following our example with colors and shapes, we supply a weighted sum as the aggregation function, i.e., the weighted sum of the color and shape descriptor distances. This means that we perceive two objects similar if they are similar in both color and shape and weights are used to emphasize either color over shape or the other way round.

3.3.5 Threshold Algorithm

The threshold algorithm [Fagin et al., 2001] (TA) was proposed for obtaining top-k results of several ranked lists. Even though it was originally proposed for the scores (ranks), it can be inverted to suit the distance measures. Let *Top(k, q, f)* be the top-k query for the query object *q* and aggregate function *f* that we want to solve on a database with *m* descriptors $s_1, \ldots, s_m$. Note that each descriptor forms a metric space with its own metric function $d_1, \ldots, d_m$. Then, the sorted access of descriptor s_i is a list S_i of all objects and their distances to the query object using only the descriptor *i*. The list is sorted by distances; the lower the distance is the sooner the object appears in the list. On the other hand, random access can retrieve the distance between the query and a given object *o* for all the descriptors and thus compute the overall distance of this object as $d(q, o) = f(d_1(q, o_1), \ldots, d_m(q, o_m))$. Then, the algorithm works as follows:

1. Do sorted access for all *m* descriptors to get their respective sorted lists.
2. In every iteration, the next object is taken from each sorted list $o_1 \in S_1, \ldots, o_m \in S_m$ having the respective descriptor's distances $\delta_1 = d(q, o_1), \ldots, \delta_m = d(q, o_m)$.
3. Use the random access to the other lists to compute the overall distances of objects $o_1, \ldots, o_m$. Update the list of the resulting k objects with the lowest overall distances. Let d^{max} be the distance of the k^{th} object, i.e. the maximal distance of the result.
4. Compute the threshold value $t = f(\delta_1, \ldots, \delta_m)$. Do next iteration unless the result list has *k* objects and $d^{max} \le t$.

The correctness of the stop condition in the fourth step is proved in the original paper [Fagin et al., 2001] and the modification for metric distances is straightforward. Basically, the threshold *t* can only increase in each iteration of the algorithm while the maximal distance d^{max} only decreases after the result list is filled with *k* objects.

3.3.6 MUFIN Implementation

To evaluate similarity queries efficiently, we build a P2P index for each of the descriptors. So, every single descriptor of a particular ContentObject[1] is stored by the respective index along with an identifier of the metaobject. Moreover, a special zero-overlay is defined where complete metaobjects are stored. The zero-overlay allows efficient retrieval of metaobjects using their identifiers as a key – a classical P2P distributed hash table can be used, because we only need the "get-by-id" operation in this overlay. In principle, these overlays are allowed to share the same infrastructure of physical peers.

[1] for a definition of ContentObject in SAPIR refer to Deliverable D2.1 – Section 4.1.2

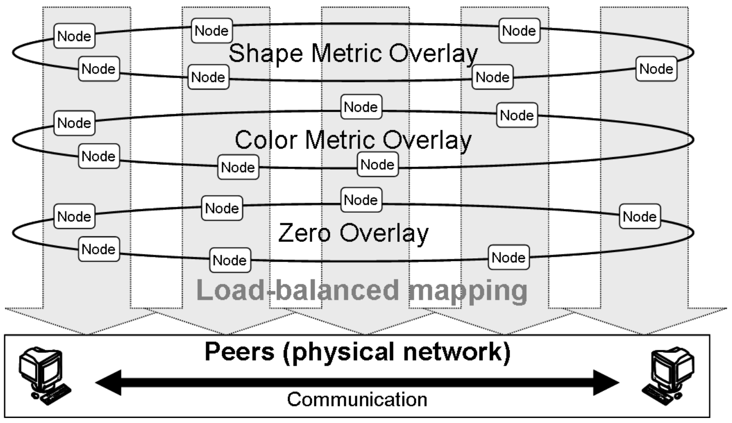


Figure 7: Multiple overlays schema

Figure 7 depicts a system with three overlays. The first is built for color descriptors, the second indexes shapes, and the third represents the zero-overlay. Observe that each overlay consists of multiple nodes and their specifics are left up to a particular distributed index structure used in the overlay. These nodes are maintained by physical peers (illustrated by the dotted arrows). Each peer usually manages one node from every overlay. Such a mapping is completely transparent for overlay index structures and in general, it is automatically done by a load-balancing mechanism.

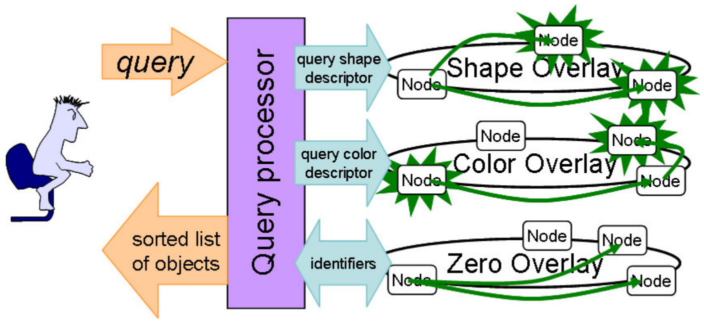


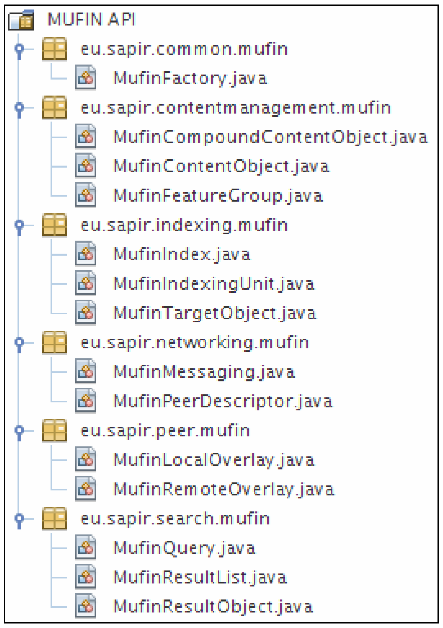
Figure 8: Evaluation of a complex query

Running a standard TA in a distributed environment would be very expensive, because single object retrieval is very inefficient. The batch approach is more suitable and it works as follows (see Figure 8 for schematic overview). The issuing peer breaks the query ContentObject into its descriptors and executes a nearest neighbor query for every descriptor in the respective similarity-search overlay. They are evaluated in parallel and a sorted list of the top-most similar objects is returned for each descriptor. The ContentObjects are then used to query the zero-overlay to get distances for missing descriptors. Next, an aggregation function is used to compute the objects' overall similarity. If there are not enough objects with their overall similarity under a certain threshold value, the descriptor overlays are requested to provide additional batch of objects until this condition is met.

However, to get under the threshold can take a lot of time for huge data collections. For example, a top-50 query in a dataset of 1.6 million images takes more than one minute to evaluate even for a batch size of 1,000 objects. Moreover, the interpretation of similarity itself is highly individual so even the optimal results of the search may not satisfy user needs. Therefore, it is more reasonable to give good-enough results quickly even if they are not precise. Thus, we alter the stop condition and we end the processing prematurely after ε iterations even if the threshold condition is not satisfied. The value ε allows us to tune the ratio of the response time and the quality of the result.

Since the parameter ε is specific for each query, the system can automatically adjust the value according to user preferences or the actual system load. To help the system tune the ε more precisely, we can compute actual quality estimations during the iterations of the TA. Since the actual threshold value t and the maximal result-list distance d^{max} are updated in every iteration of TA, we can see them as functions $t(i)$ and $d^{max}(i)$ of the TA iteration i. If we know the final maximal distance $d^{max}(final)$ of the precise result, we can express the quality of the result after i iterations as a ratio $d^{max}(i)/d^{max}(final)$. The best quality is equal to 1 (precise result) while the higher values represent worse quality. However, the final distance is unknown during the evaluation and we can only use the threshold $t(i)$ as its lower bound (due to the TA stop condition). The quality is therefore upper-bounded by $d^{max}(i)/t(i)$. Using the first ε values of $t(i)$ and $d^{max}(i)$, we can improve the estimation of the quality by extrapolating the behavior of the $t(i)$ and $d^{max}(i)$ functions. Then, their intersection can be computed, and the function value at the intersection is an estimation of $d^{max}(final)$ and can be used to compute the estimated quality at iteration ε.

3.4 MUFIN IN SAPIR



```
MUFIN API
  eu.sapir.common.mufin
    MufinFactory.java
  eu.sapir.contentmanagement.mufin
    MufinCompoundContentObject.java
    MufinContentObject.java
    MufinFeatureGroup.java
  eu.sapir.indexing.mufin
    MufinIndex.java
    MufinIndexingUnit.java
    MufinTargetObject.java
  eu.sapir.networking.mufin
    MufinMessaging.java
    MufinPeerDescriptor.java
  eu.sapir.peer.mufin
    MufinLocalOverlay.java
    MufinRemoteOverlay.java
  eu.sapir.search.mufin
    MufinQuery.java
    MufinResultList.java
    MufinResultObject.java
```

Figure 9: MUFIN API, packages and classes

3.4.1 Introduction

In Figure 9 we report the packages of the MUFIN API and the classes. A javadoc for MUFIN is given as an appendix of this document.

The MUFIN implementation was used in SAPIR for experimenting large scale indexing of photos. In SAPIR we built and used the CoPhIR[2] collection. CoPhIR is the largest publicly available collection of high-quality images metadata. It consist of 106 Million images. Each contains five MPEG-7 visual descriptors (Scalable Color, Color Structure, Color Layout, Edge Histogram, Homogeneous Texture), and other textual information (title, tags, comments, etc.) of photos that have been crawled from the Flickr photo-sharing site.

[2] http://cophir.isti.cnr.it

Since no collection of this scale was available for research purpose, we had to tackle the non-trivial process of image crawling and descriptive feature extraction using the European EGEE computer GRID. In particular, we had the possibility to access the EGEE (Enabling Grids for EsciencE) European GRID infrastructure provided to us by the DILIGENT IST project.

More details about the crawling process can be found in Deliverable D4.5.

3.4.2 CoPhIR and MUFIN in SAPIR

The CoPhIR metadata built and used to perform experiments in the SAPIR project, contains five MPEG-7 visual descriptors:

Scalable Colour

It is derived from a colour histogram defined in the HueSaturation-Value colour space with fixed colour space quantization. The histogram values are extracted, normalized and nonlinearly mapped into a four-bit integer representation. Then the Haar trasform is applied. We use the 64 coefficients version of this descriptor.

Colour Structure

It is also based on colour histograms but aims at identifying localized colour distributions using a small structuring window. We use the 64 coefficients version of this descriptor.

Colour Layout

It is obtained by applying the DCT transformation on a 2-D array of local representative colours in Y or Cb or Cr colour space. This descriptor captures both colour and spatial information. We use the 12 coefficients version of this descriptor.

Edge Histogram

It represents local-edge distribution in the image. The image is subdivided into 4×4 sub-images, edges in each sub-image are categorized into five types: vertical, horizontal, 45° diagonal, 135° diagonal and nondirectional edges. These are then transformed in a vector of 80 coefficients.

Homogeneous Texture

It characterizes the region texture using the mean energy and the energy deviation from a set of 30 frequency channels. We use the complete form of this descriptors which consist of 62 coefficients.

More information about these MPEG-7 Visual Descriptors can be found in Deliverable 3.1.

The metric space model adopted by SAPIR, provides us with a unique generality and almost absolute freedom in designing and combining the similarity measures. Specifically, as the individual MPEG-7 features and their distance functions form metric spaces, we can combine several features into a single metric function by a weighted sum of the individual feature distances. The particular distances are normalized before being weighted and summed.

For indexing the CoPhiR dataset using MUFIN in SAPIR we do not use the dynamic aggregate function that was described in Section 3.3. The CoPhIR dataset was indexed using MUFIN merging the MPEG-7 visual descriptors together and using the weighted sum of the individual distances as global distance. The following table summarizes the features we use, their respective distance measures (i.e., the ones suggested by the MPEG group [Manjunath et al. 2002]), and the weights used in the aggregate metric function. These weights were determined performing experiments and considering the work reported in [Amato et al. 2004].

MPEG-7 Feature	**Metric**	**Weight**
Scalable Color	L_1 metric	2
Color Structure	L_1 metric	3
Color Layout	sum of L_2	2
Edge Histogram	special	4
Homogeneous Texture	special	0.5

4 SUMMARY

In this report we have presented the result of the work done in SAPIR about executing complex similarity queries over multi layer P2P search structures which was the main topic of Task T5.4. In particular we reported the algorithms for similarity search complex queries execution and the implementation of the MUFIN overlay.

This document is accompanied by a zip file containing a javadoc for MUFIN.

REFERENCES

Aberer Karl Efficient Search in Unbalanced, Randomized Peer-To-Peer Search Trees [Report] : Technical Report / Ecole Polytechnique Fédérale de Lausanne (EPFL). - Lausanne : [s.n.], 2002.

Amato G., Falchi F., Gennaro C., Rabitti F., Savino P., Stanchev P., Selection of MPEG-7 Image Features for Improving Image Similarity Search on Specific Data Sets // Proceedings of the 7-th IASTED International Conference on Computer Graphics and Imaging (CGIM 2004), ACTA Press (Calgary, Canada, 2004): 395-400.

Amato G., Falchi F., Gennaro C., Rabitti F., Savino P., Stanchev P., Improving Image Similarity Search Effectiveness in a Multimedia Content Management System // MIS 2004, 10th International Workshop on Multimedia Information Systems, August 25-27, 2004, University of Maryland, College Park, MD, USA, Position Papers: 139-146.

Aspnes, J. and Shah, G., Skip graphs. *ACM Trans. Algorithms* 3, 4 (Nov. 2007), 37

Balakrishnan Hari [et al.] Looking up data in P2P systems. [Journal] // Communications of the ACM. - [s.l.] : ACM Press, February 2003. - 2 : Vol. 46. - pp. 43-48.

Batko Michal [et al.] On Scalability of the Similarity Search in the World of Peers [Conference] // InfoScale '06: proceedings of the First International Conference on Scalable Information Systems. - New York : ACM Press, 2006. - p. 20.

Batko Michal, Gennaro Claudio and Zezula Pavel Similarity Grid for Searching in Metric Spaces. [Conference] // Peer-to-Peer, Grid, and Service-Orientation in Digital Library Architectures. 6th Thematic Workshop of the EU Network of Excellence DELOS, Cagliari, Italy, June 24-25, 2004, Revised Selected Papers. - [s.l.] : Springer-Verlag Berlin Heidelberg, 2004. - Vols. Lecture Notes in Computer Science, 3664. - pp. 25-44.

Batko Michal, Kohoutkova Petra and Zezula Pavel Combining Metric Features in Large Collections. In Proceedings of the 1st International Workshop on Similarity Search and Applications (SISAP 2008). 2008. - pp. 79-86.

Batko Michal, Novak David and Zezula: Pavel MESSIF: Metric Similarity Search Implementation Framework [Conference] // Digital Libraries: Research and Development. First International DELOS Conference, Pisa, Italy, February 2007, Revised Selected Papers. - [s.l.] : Springer Berlin / Heidelberg, 2007. - pp. 1-10.

Batko Michal, Novák David e Zezula Pavel MESSIF: Metric Similarity Search Implementation Framework [Atti di convegno] // Conference 2007: Working Notes, Pisa, 13-14 February 2007. Pisa, Italy. - [s.l.] : Information Society Technologies, 2007. - p. 11-23. - ISBN 2-912335-30-2.

Bharambe A. R., Agrawal M. and Seshan S. Mercury: Supporting scalable multi-attribute range queries [Conference] // ACM SIGCOMM Computer Communication Review. - [s.l.] : ACM, 2004. - Vol. 34 (4). - pp. 353-366.

Böhm Christian, Berchtold Stefan and Keim Daniel A. Searching in high-dimensional spaces: Index structures forimproving the performance of multimedia databases [Journal] // ACM Computing Survreys (CSUR). - [s.l.] : ACM, September 2001. - 3 : Vol. 33. - pp. 273-321.

Bozkaya Tolga and Ozsoyoglu Meral Indexing large metric spaces for similarity search queries. [Journal] // ACM Transactions on Database Systems (TODS). - [s.l.] : ACM Press, September 1999. - 3 : Vol. 24. - pp. 361-404.

Broder A. J. Strategies for efficient incremental nearest neighbor search [Journal] // Pattern Recognition. - [s.l.] : Elsevier Science Inc., January 1990. - 1-2 : Vol. 23. - pp. 171-178.

Ciaccia P., Patella M., and Zezula P., M-Tree: An efficient access method for similarity search in metric spaces. In Proceedings of 23rd International Conference on Very Large Data Bases (VLDB'97), August 25-29, 1997, Athens, Greece, pages 426–435, 1997.

Ciaccia P., Patella M., and Zezula P., Processing Complex Similarity Queries with Distance-Based Access Methods. In Hans-Jörg Schek, Fèlix Saltor, Isidro Ramos, and Gustavo Alonso, editors, EDBT, volume 1377 of Lecture Notes in Computer Science, pages 9–23. Springer, 1998. ISBN 3-540-64264-1.

Cai M. [et al.] MAAN: A multiattribute addressable network for grid information services [Conference] // GRID '03: Proceedings of the Fourth InternationalWorkshop on Grid Computing. - [s.l.] : IEEE Computer Society, 2003. - pp. 184-191.

Chávez Edgar, Navarro Gonzalo and Baeza-Yates Ricardo Searching in metric spaces. [Journal] // ACM Computing Surveys (CSUR). - September 2001. - 3 : Vol. 33. - pp. 273–321.

Clarkson Kenneth L. Nearest neighbor queries in metric [Conference] // STOC '97: Proceedings of the twenty-ninth annual. - El Paso, Texas, United States : ACM Press, 1997. - pp. 609-617.

Dohnal Vlastislav [et al.] D-Index: Distance Searching Index for Metric Data Sets [Journal]. - [s.l.] : Springer Netherlands, September 2003. - 1 : Vol. 21. - pp. 9-33.

Fagin R., Combining Fuzzy Information from Multiple Systems // Proceedings of the Fifteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 3-5, 1996, Montreal, Canada, pages 216–226. ACM Press, 1996. ISBN 0-89791-781-2.

Fagin R., Lotem A., and Naor M. Optimal aggregation algorithms for middleware. In Proceedings of the 20th ACM PODS'01, 2001. - pp. 102-113.

Fabrizio F., Gennaro Claudio and Zezula Pavel A Content-Addressable Network for Similarity Search in Metric Spaces. [Conference] // DBISP2P '05: Proceedings of the the 2nd International Workshop on Databases, Information Systems. - [s.l.] : Springer, 2005. - Vol. 4125 of Lecture Notes in Computer Science. - pp. 98-110.

Fabrizio F., Gennaro Claudio and Zezula Pavel Nearest Neighbor Search in Metric Spaces through Content-Addressable Networks [Journal] // Information Processing and Management: an International Journal. - [s.l.] : Pergamon Press, Inc., January 2008. - 1 : Vol. 44.

Falchi F., Gennaro C., Rabitti F., Zezula P., A Distributed Incremental Nearest Neighbor Algorithm, // Proceedings of the 2nd international conference on Scalable information systems (Infoscale'07), ISBN:978-1-59593-757-5, ICST (Brussels, Belgium, Belgium), 2007, Article No 82.

Falchi F., Gennaro C., Rabitti F., Zezula P., Distance Browsing in Distributed Multimedia Databases. // Future Generation Computer Systems, vol. 25, Issue 1 (January 2009), Elsevier Science Publishers B. V. (Amsterdam, The Netherlands), 2009: 64-76.

Foster Ian T. and Iamnitchi Adriana On Death, Taxes, and the Convergence of Peer-to-Peer and Grid Computing. [Conference] // IPTPS 2003: Peer-to-Peer Systems II, Second International, Revised Papers. - [s.l.] : Springer, 2003. - Vol. Lecture Notes in Computer Science 2735. - pp. 118-128.

Ganesan P., Yang B. and Garcia-Molina H. One torus to rule them all: Multidimensional queries in P2P systems [Conference] // WebDB '04: Proceedings of the 7th International Workshop on the Web and Databases. - [s.l.] : ACM Press, 2004. - pp. 19-24.

Goldstone Robert L. and Yun. Son Ji [Book Section] // Cambridge Handbook of Thinking and Reasoning / ed. Morrison K. Holyoak and R.. - Cambridge : Cambridge University Press, 2005.

Hjaltason G´ısli R. and Samet Hanan Distance Browsing in Spatial Databases [Journal] // ACM Transactions on Database Systems (TODS). - [s.l.] : ACM Press, 1999. - 2 : Vol. 24. - pp. 265-318.

Hjaltason Gisli R. and Samet Hanan Index-driven similarity (Survey Article). [Journal] // ACM Transactions on Database Systems (TODS). - [s.l.] : ACM Press, December 2003. - 4 : Vol. 28. - pp. 517-580.

James William The Principles of Psychology. [Book]. - [s.l.] : Dover, 1890/1850.

Kelley John L. General Topology [Book]. - New York Heidelberg Berlin : Van Nostrand Reinhold, 1955.

Litwin Witold, Neimat Marie-Anne and Schneider Donovan A. LH* - A Scalable, Distributed Data Structure. [Journal] // ACM Transactions on Database Systems (TODS). - [s.l.] : ACM Press, 1996. - 4 : Vol. 21. - pp. 480-525.

Litwin Witold, Neimat Marie-Anne and Schneider Donovan A. LH* - Linear Hashing for Distributed Files. [Conference] // ACM SIGMOD Record. - [s.l.] : ACM Press, 1993. - Vol. 22 (2). - pp. 327-336.

Litwin Witold, Neimat Marie-Anne and Schneider. Donovan A. RP*: A Family of Order Preserving Scalable Distributed Data Structures. [Conference] // In VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile. - [s.l.] : Morgan Kaufmann, 1994. - pp. 342-353.

Manjunath, B., Salembier, P., Sikora, T. (eds.): Introduction to MPEG-7: Multimedia Content Description Interface. John Wiley & Sons, Inc., New York, NY, USA (2002)

Novak David and Zezula Pavel M-Chord: a scalable distributed similarity search structure. [Conference] In InfoScale '06: Proceedings of the 1st international conference on Scalable information systems. - [s.l.] : ACM Press. - p. 19. 2006.

Samet Hanan Foundations of Multidimensional and Metric [Book]. - San Francisco : Morgan Kaufmann Publishers Inc., 2006.

Shirky Clay Listening to Napster. [Book Section] // Peer-to-Peer: Harnessing the Power of Disruptive Technologies. / ed. Oram Andy. - Sebastopol : O'Reilly & Associates, Inc., 2001.

Steinmetz Ralf and Wehrle Klaus Peer-to-Peer-Networking & -Computing. [Journal] // Informatik Spektrum. - 2004. - 1 : Vol. 27. - pp. 51-54.

Steinmetz Ralf and Wehrle Klaus What Is This Peer-to-Peer About? [Book Section] // Peer-to-Peer Systems and Applications. - [s.l.] : Springer-Verlag Berlin Heidelberg, 2005. - Vol. Lecture Notes in Computer Science 3485.

Steinmetz Ralf and Wehrle, Klaus Peer-to-Peer Systems and Applications [Book]. - [s.l.] : Berlin Heidelberg, Germany, 2005. - Vol. 3485 of Lecture Notes in Computer Science.

Stoica Ioan, Morris Robert, Karger David R., Kaashoek M. Frans, Dabek Frank, and Balakrishnan Hari. Chord: A scalable peer-to-peer lookup service for internet applications. [Conference] // SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications. - [s.l.] : Acm Press, 2001. - pp. 149-160.

Wehrle Klaus, Gtz Stefan and Rieche Simon Distributed Hash Tables [Book Section] // Peer-to-Peer Systems and Applications. - [s.l.] : Springer-Verlag Berlin Heidelberg, 2005. - Vols. Lecture Notes in Computer Science, 3485.

Zezula Pavel [et al.] Similarity Search. The Metric Space Approach [Book]. - New York : Springer Science + Business Media, Inc., 2006.