Evaluation of Similarity Searching Methods for Music Data in Peer-to-Peer Networks

Ioannis Karydis, Alexandros Nanopoulos, Apostolos N. Papadopoulos and Yannis Manolopoulos

Data Engineering Lab., Department of Informatics
Aristotle University, 54124, Thessaloniki, GREECE
{*karydis,alex,apostol,manolopo*}*@delab.csd.auth.gr*

Abstract

In this paper, we focus on similarity searching for similar acoustic data over unstructured decentralized P2P networks. Similarity is measured in terms of time warping, which can cope with distortion that is naturally present when query-by-content is performed. We propose a novel framework, which takes advantage of the absence of overhead in unstructured P2P networks and minimizes the required traffic for all operations with the use of an intelligent sampling scheme. Within the proposed framework we adapt several existing algorithms for searching in P2P networks. A detailed experimental evaluation which compares the performance of the various similarity searching algorithms demonstrates the efficiency of the proposed scheme.

Keywords: music data, multimedia databases, similarity searching, peer-to-peer networks, data mining.

1 Introduction

The World Wide Web (WWW) is being used for commercial, entertainment or educational purposes, and has become the primary means for information dissemination. One popular type of data that is being disseminated over WWW is digitized music. Recently, the new opportunities that emerge from this activity have been recognized and led to the development of systems like iTune (www.apple.com/itunes, iMusic (www.imusic.com), and Napster (www.napster.com). Although abundantly used, traditional metadata (title, composer, performer, genre, date, etc.) of a music object give rather minimal information regarding the actual content of the music object itself. On the other hand, research efforts in the field of Music Information Retrieval (MIR) have developed efficient methods for searching music data collections *by content*. For instance, queries based on humming (using a microphone) or on a small piece of musical file, are a more natural approach to MIR. This type of queries lies within the Content-Based MIR (CBMIR). In CBMIR, an actual music piece is required in order to compare its content with the content of the music pieces already available in the database.

As with regards to the infrastructure for exchanging music data, peer-to-peer (P2P) networks over the WWW have gain significant popularity recently. Within the advantageous qualities of the P2P networks lies the increased size of the overall database offered by a P2P network, its fault tolerance support to peer failure by other peers and the workload distribution over a network of available CPUs, since CBMIR is computationally highly intensive. Nonetheless, the very advantages of the P2P network are the same parameters that make P2P information retrieval much more complex than the traditional search methods. That is, the lack of a central repository for the documents to be retrieved, the large number of documents available and the dynamic character of the network, introduce an increased degree of difficulty in the retrieval process.

P2P networks can be classified based on the control over data location and network topology in *unstructured*, *loosely structured* and *highly structured* [13]. The absence of structure allows for resilience in dynamic environments (peer join/leave) while no guaranties can be given on the retrieval of existing documents. Moving towards increased structure, both the probability of retrieving existing documents and the overhead of handling peer join-leave augment. Additionally, P2P networks can also be classified according to the number of central directories of document locations in *centralized*, *hybrid* and *decentralized*. Centralized P2P networks are subject to the same drawbacks for which the traditional server-client model was originally abandoned (network failures due to central peer failure, impaired scalability, joining/leaving of peers not easily handled, possible undesirable dominion of controllers). For these reasons, we focus on decentralized unstructured P2P networks, which overcome the aforementioned drawbacks. The absence of structure was selected for the looseness of control over the data location, that is each peer can share its own documents without hosting any documents of other peers due to locality restrains.[1]

Searching for music data by content requires the development of effective and efficient similarity searching methods for this kind of data, which will find music files that are similar to a query music datum. Moreover, the particularities of decentralized unstructured P2P networks have to be taken into account in order to lead to feasible solutions. For instance, the traffic that similarity queries pose over the network is crucial. Additionally, the amount of processing required in each peer is an important factor. For these reasons, what is needed is (i) a model that will effectively capture the similarity between music data and will be also fast to compute, and (ii) efficient similarity searching algorithms, which will minimize the traffic.

In this paper, we utilize the Dynamic Time Warping (DTW) as the similarity measure between two musical pieces. A significant characteristic of DTW is its ability to withstand distortion of the comparing series in the time axis, a property that is suitable for effective searching for similar acoustic data (e.g., wav, mp3), which can be represented as time series (that record signal amplitude over time). Since different performances of the same musical piece may include locally differentiated tempo, DTW seems a natural choice for this problem [11]. For this reason, it has been recently proposed for shake of MIR in centralized environments [25, 14, 6, 1]. Moreover, for the fast computation of the DTW measure, we consider lower-bounding techniques that have been proposed in the research field of time-series databases. Based on DTW, we develop a framework for developing similarity searching algorithms, which is based on appropriate sampling and representation methods for music data, and takes into account the requirements of P2P networks. In this framework we examine some recently proposed algorithms for searching similar text documents in P2P networks. The considered algorithms are adapted in order to fit the requirements of the examined paradigm.

The technical contributions of this paper are summarized as follows:

- The development of a novel framework for efficient retrieval of audio data similar to an audio query in an decentralized unstructured P2P network. The proposed framework significantly reduces the required traffic for all operations with the use of an intelligent sampling scheme on the lower and upper bounds used. The proposed algorithm has such a design that no false negative results occur.

- The examination of algorithms that use selective criteria in order to efficiently search over the P2P network.

[1]We must notice that with the examined framework we refer to applications that support content sharing for legal subscribers (e.g., iTunes, iMusic). Moreover, it is interesting to notice that the proposed approach can be adopted as a means of identification of illegal sharing, by finding sites that share unregistered content.

- The detailed experimental results which show the efficiency of the proposed framework and the comparative performance of the different searching algorithms.

Taking into consideration that similarity searching is an essential part of clustering algorithms in data mining applications [15], the proposed method could be used as an efficient similarity search component towards clustering of music data in a P2P environment.

The rest of the paper is organized as follows. Section 2 describes related work. In Section 3, we describe the similarity model that is based on DTW. Section 4 provides a complete account of the framework proposed in this paper, whereas Section 5 presents the similarity searching algorithms that are used in this framework. Subsequently, Section 6 presents and discusses the experimentation and results obtained. Finally, the paper is concluded in Section 7.

2 Related work

2.1 Summary of existing P2P systems

P2P networks can be classified based on the control over data location and network topology in *unstructured*, *loosely structured* and *highly structured* [13]. Unstructured P2P networks follow no rule in where data is stored while the network topology is arbitrary (e.g., Gnutella). Loosely structured P2P networks have both data location and network architecture non-precisely determined (e.g., Freenet). Finally, in highly structured networks data storage and network topology are explicitly defined (e.g., Chord). What is more, P2P networks can also be classified according to the number of central directories of document locations in *centralized*, *hybrid* and *decentralized.* Thus, centralized networks maintain a central directory in a single location (e.g., Napster), hybrid networks maintain more than directories in super-peers (e.g., Kazaa) while for the decentralized (e.g., Chord) no central directory is kept. P2P networks can also be classified into hierarchical and non-hierarchical based on whether their overlay structure is a hierarchy or not. It is common for decentralized systems to have no hierarchy, while hybrid and most centralized systems ordinarily incorporate some degree of hierarchy. Hierarchical systems provide increased scalability, ease in exploiting peer heterogeneity and high routing efficiency. On the other hand systems with no hierarchy offer load-balance and increased resilience.

In this work we define our network to be a decentralized, unstructured, non-hierarchical network. Additionally, we assume that the system consists of N peers, each peer having approximately $\log N$ neighboring peers. The diameter of the system is the maximum number of peers required as intermediate steps in order to reach a peer P_i from a peer P_j. We define MaxHop to be the number of peers a query should be forwarded to. The MaxHop of a query is initially set to a value and each time the query is forwarded to a peer, MaxHop is reduced until reaching zero, from which point on the query is not propagated further. The MaxHop parameter is equivalently called Time To Leave (TTL).

2.2 Searching methods in unstructured P2P networks

In this section we summa rise a number of different searching methods for decentralized unstructured P2P networks. Initially, we examine the Breadth-First Search (BFS) algorithm. In the BFS, a query peer Q propagates the query q to all its neighbor peers. Each peer P receiving the q initially searches its local repository for any documents matching q and then passes on q to all its neighbors. In case a P has a match in its local repository then a *QueryMatch* message is created containing information about the match. The *QueryMatch* messages are then transmitted back, using reversely the path q travelled, to Q. Finally, since more than one *QueryMatch* messages have been received by Q, it can select the peer with best connectivity attributes for direct downloading of the match. It is obvious that the BFS sacrifices performance and network traffic for simplicity and high-hit rates. In order to reduce network traffic, the TTL parameter is used (see Section 2). In a modified version of this algorithm, the Random BFS (RBFS) [7], the query peer Q propagates the query q not to all but at a fraction of its neighbor peers.

In an attempt to rectify the inability of the RBFS to select a path of the network leading to large network segments, the $>RES$ algorithm was developed [20]. In this approach, the query peer Q propagates the query q to a subset of its neighbor peers based on an aggregated statistic. That is, Q propagates the q to k neighboring peers, all of which returned the highest number of results during the last m queries, with k and m being configurable parameters. $>RES$ is a significant amelioration in comparison to the RBFS algorithm, however its attitude is rather quantitative than qualitative, since it does not select the neighbors to propagate the query q based on the similarity of the content of q with the previous queries.

To overcome this quantitative behavior of $>RES$ approach, *ISM* has been proposed [7]. In *ISM*, for each query, a peer propagates the query q to the peers that are more likely to reply the query based on the following two parameters; a profile mechanism and a relevance rank. The profile is built and maintained by each peer for each of its neighboring peers. The information included in this profile consists of the t most recent queries with matches and their matches as well as the number of matches the neighboring peer reported. The relevance rank (RR) function is computed by comparison of the query q to all the queries for which there is a match in each profile. Thus for a querying peer P_Q the RR function is calculated by the following formula:

$$RR_Q(P_i, q) = \text{Qsim}(q_j, q)^{\alpha} \times S(P_i, q_j)$$

where Qsim is the similarity function used between queries and S(P_i,q_j) is the number of the results returned by P_i for query q_j. The *ISM* allows for higher ranking of the neighboring peers that return more results by adjustment of the α parameter. Obviously, the strong point of the *ISM* approach is in environments that show increased degree of document locality.

2.3 MIR in P2P networks

The field of combined CBMIR and P2P networks is definitely very young as the inaugural research paper dates back in 2002 [18]. Despite, the limited number of works that exist, are presented thereinafter.

In this first attempt, the authors of [18] present four P2P models for CBMIR. The four models include all centralized, decentralized and hybrid categories. Accordingly, the authors of [18] propose a retrieval acceleration algorithm based on difference in pitch between two tones of music and a result filtering method relying on replication removal techniques. Additionally, the authors propose an architecture of a CBMIR P2P system, that falls within the hybrid category of P2P systems.

Another research based on a hybrid configuration is presented in [17]. Therein the authors propose a system that utilizes both manually specified attributes (artist, album, title, etc.) and extracted features in order to describe the musical content of a piece. The underlying P2P network is a DHT-based system. In such systems each node is assigned with a region in a virtual address space, while each shared document is associated with a value of this address space. Thus, locating a document requires only a key lookup of the node responsible for the key.

The author in [22] proposed the utilization of the feature selection and extraction process that is described in [21] for CBMIR in a decentralized unstructured P2P system. The research considers both a replicated database and a general P2P scenario, while special attention is given on the control of the workload produced at queried peers during query time. Each query is divided into two phases, the first of which includes only a subpart of the actual query vectors, in order to distinguish high probability response peers. Accordingly, a peer ranking occurs and the full query vectors are sent to all peers. Given that a peer has free CPU resources, it decides whether to process a query or not based on the ranking that the specific query received. It is obvious that this approach produces large network traffic, since the full query vectors are sent to all peers, instead of the most promising.

Finally, although oriented towards a differentiated discipline, the work of Shrestha et al. [16] refers to audio retrieval in P2P networks. The principal target of this research is combating of unauthorized music file sharing in P2P networks. To achieve this, peers are divided into hierarchical groups according to their resources, while each task is assigned to several peers to overcome the dynamic peer join-leave.

3 Similarity model based on DTW

The efficient processing of similarity queries requires addressing of the following important issues:

- the definition of a meaningful distance measure $D(S, C)$ in order to express the similarity between two time series objects S and C,
- the efficient representation of time series data, and
- the application of an appropriate indexing scheme in order to quickly discard database objects that can not contribute to the final answer.

One of the most fundamental research issues in time series is the definition of meaningful measures towards time series similarity expression. Given two time series S and C the problem is to define a distance measure $D(S, C)$ which expresses the degree of similarity between S and C. One of the most widely used distance measures for time series is the Euclidean distance (L_2 norm), which has the restriction that both series must be of the same length. Given two time series S and C of length N, the Euclidean distance is defined as follows:

$$D_{euclidean} = \sqrt{\sum_{i=1}^{N}(S_i - C_i)^2} \quad (1)$$

where S_i, C_i are the value of S and C for the i-th time instance. The Euclidean distance has been widely used as a similarity measure in time series literature [2, 5, 4, 12], due to its simplicity.

Several alternative distance functions have been proposed in order to allow translation, rotation and scaling invariance [3, 24, 4, 23]. Consider for example the time series depicted in Figure 1. Note that although all time series have the same shape, they will be considered non-similar if the Euclidean distance is used to express similarity.

Taking into consideration that the Euclidean distance does not always meet the application's requirements, Dynamic Time Warping (DTW) has been proposed as a more robust similarity measure. DTW can express similarity between two time series even if they are out of phase in the time axis, or they do not have the same length. The DTW distance $D_{DTW}(S, C)$ between time series S and C is essentially a way to map S to C and vice-versa. This process is also known as *alignment* of time series. If S is of length N and C is of length M, then the distance D_{DTW} can be evaluated by using the following method:

1. An $N \times M$ matrix is constructed, where the cell in the i-th row and the j-th column contains the distance $d(S_i, C_j) = (S_i - C_j)^2$.
2. A warping path is defined which is a contiguous set of matrix cells that defines a mapping between elements of S and elements of C.

Although there are many warping paths that map S to C, what is required is to determine the most promising one, by trying to optimize the cumulative distance $\gamma(i,j)$ in each cell of the warping path. Therefore, the following recurrence is defined:

$$\gamma(i,j) = d(S_i, C_j) + min\{\gamma(i-1,j-1), \gamma(i-1,j), \gamma(i,j-1)\} \tag{2}$$

Figure 1 illustrates an example of two time series aligned by means of the Euclidean distance (Figure 1(a)) and by DTW distance (Figure 1(b)). It is evident that the two time series are similar but their phases are different. However, their similarity can not be captured by the Euclidean distance.

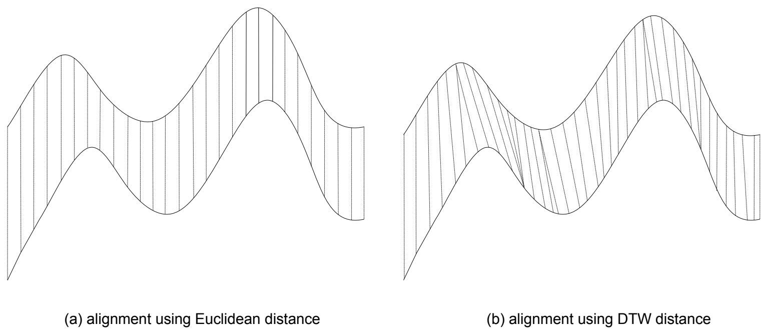

(a) alignment using Euclidean distance
(b) alignment using DTW distance

Figure 1: Time series alignment with Euclidean and DTW distances.

The most important disadvantage of the DTW method is that it does not satisfy the triangular inequality, which is a desirable property for constructing efficient indexing schemes and pruning the search space. Moreover, the computation of $D_{DTW}(S,C)$ is significantly more CPU intensive than that of $D_{Euclidean}(S,C)$. Therefore, an interesting direction for performance improvement is the definition of a lower bound, in order to take advantage of indexing schemes and avoid the computation of DTW when there is a guarantee that the two time series are not similar. In this work we utilize the lower bound proposed in [9] which is termed LB_Keogh. For a sequence S, LB_Keogh is based on the *envelope* of S, which consists of the upper, U, and lower, L, sequences. Given a parameter r (which is denoted as *reach* and confines the search space of DTW – see [9] for more details), the i-th elements of U and L are defined as follows:

$$U_i = \max(S_{i-r} : S_{i+r})$$

$$L_i = \min(S_{i-r} : S_{i+r})$$

Essentially, for each i, the upper bound guarantees that $U_i \geq S_i$ and the lower bound guarantees that $L_i \leq S_i$. A schematic illustration of U and L is given in Figure 2.

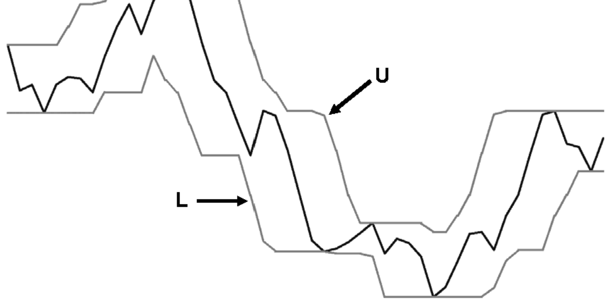


Figure 2: Illustration of U and L sequences.

Given U and L for a sequence S, the value of LB_Keogh between S and a sequence C is defined as follows:

$$LB\_Keogh(S, C) \quad = \quad \sqrt{\sum_{i=1}^{N} \begin{cases} (C_i - U_i)^2, & \text{if } C_i > U_i \\ (C_i - L_i)^2, & \text{if } C_i < L_i \\ 0, & otherwise \end{cases}} \qquad (3)$$

Clearly, the computation of LB_Keogh is linear to the size of the sequences and is much faster than computing the actual D_{DWT}. In [9] is proven that $LB\_Keogh(S, C) \leq D_{DTW}(S, C)$, thus $LB\_Keogh(S, C)$ can be used for pruning without producing false negatives. The effectiveness of the bound depends on how close its value is to the actual DTW distance. An example of computation for LB_Keogh between a sequence S (represented by its U and L sequences) and a sequence C is illustrated in Figure 3. The parts of C that actually contribute to LB_Keogh are those that are either above U or bellow L, and are depicted with gray color. Evidently, the larger the number of these part is, the more effective pruning is attained by LB_Keogh.

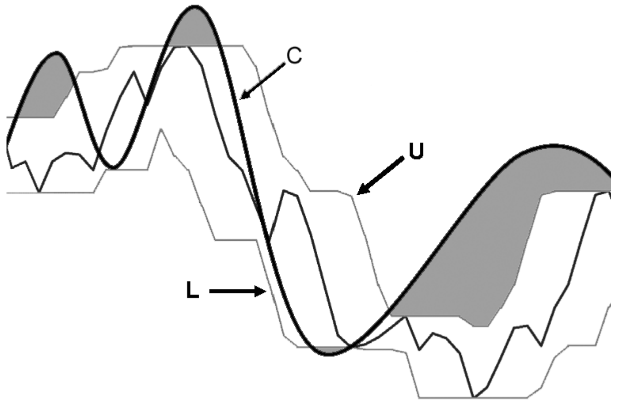


Figure 3: Example of LB_Keogh computation.

4 Proposed framework

4.1 Overview

As already mentioned, P2P searching algorithms are based on the following scheme: the node that poses the query examines its contents and finds documents that satisfy the query. Then, it selects a subset of its peers and propagates the query to them. Each peer in its turn examines its contents to find qualifying documents, and then propagates the query to a subset of its peers. To avoid the involvement of a prohibitively large number of nodes, the propagation of queries is restrained by a MaxHop parameter, which determines the number of peers a query should be forwarded to.

Due to the selected similarity model, the information that is propagated between nodes comprises the U and L sequences of the query sequence (i.e., the envelope of the query). A node that receives these sequences computes the LB value between its documents and the envelope. When a LB value is smaller than the user-specified similarity threshold, then the actual query sequence is propagated to this node[2] and the actual DTW distance is computed between the query and the corresponding document.

The queries we consider constitute music phrases, that is, excerpts of the music documents that are a type of units of music information[3]. This holds especially in the context of query by

[2]The query can be directly propagated from the node that initially posed the query, since the currently visited node always knows the address of this initial node.

[3]A minimum-length portion of the musical piece that is meaningfully independent and complete within a piece of music.

humming, where users tend to hum a piece that is (i) relatively short, (ii) well identified and separated within a song. The identification of phrases can be done following the methodology presented in [25]. In particular, a transcription algorithm [10] can produce the pitch information of the acoustic sequence. Time intervals, corresponding to phrases in the pitch information, are detected in between the time instances that silence exists (the same time intervals produce the phrases in the corresponding acoustic sequence). In summary, we are interested in finding music documents that contain phrases similar to the query sequence. Similarity through DTW is suitable in this context, since the properties of DTW help in alleviating errors that humming produces.

An important observation is that acoustic data tend to be very large. Although queries are music phrases (i.e., parts of the music sequences), the number of elements in a phrase of even few seconds can be several hundred thousands. The length of the U and L sequences is equal to the length of the query sequence. This means that a straightforward approach, which directly propagates U and L sequences between nodes, will result into an extremely large traffic over the P2P network. Moreover, when the length of the envelope's sequences are large, the computation of LB in each node can become rather costly. This violates the need of a P2P network to burden the participating nodes as little as possible. Notice that the aforementioned requirements are not present in other contexts, like the searching of similar text documents over a P2P network, where queries consist of up to some tenths of terms.

We propose a two-fold scheme which significantly reduces the traffic over the P2P network when querying music documents by content. The scheme works as follows:

- It reduces the length of the envelope's sequences by sampling them. However, plain sampling can be ineffective, since it leads to underestimation of LB. For this reason, we describe a novel sampling method to reduce the length of the sequences without significantly affecting the computation of LB. Additionally, we are interested in not introducing false-negatives due to the use of sampling.

- It uses (whenever possible) a compact representation of the sampled sequences of the envelope. The representation comprises a kind of compression for the sequences, but it does not burden the nodes of the P2P network with the cost of decompression. If the latter is not undesirable, further compression can be achieved through the use of existing methods. We do not explore this direction, since it does not affect the relative performance of the proposed scheme against the plain one that directly propagates the envelope (i.e., the performance of both methods will be equally improved).

In the following we describe the aforementioned issues in more detail.

4.2 Sampling and representation methods

Let the considered phrase length be equal to N. The length of each query Q, and therefore of its upper (U) and lower (L) sequences, will also be equal to N. We would like to sample U and L, so as to obtain two sequences U' and L', each of length $M \ll N$. Initially, we assume that uniform sampling is performed. In this case, we simply select each time the $(i \times N/M)$-th element of U and L, where $1 \leq i \leq M$. When we compute the LB_Keogh between the query sequence Q and a data sequence, we consider each phrase C of length N in Q. Each phrase has to be sampled in the same way as U and L. This leads to a sampled phrase C'. Therefore, we get a lower-bound measure LB', given as:

$$LB' = \sqrt{\sum_{i=1}^{M} \begin{cases} (C'_i - U'_i)^2, & \text{if } C'_i > U'_i \\ (C'_i - L'_i)^2, & \text{if } C'_i < L'_i \\ 0, & \textit{otherwise} \end{cases}} \qquad (4)$$

In the aforementioned equation, the third case (i.e., when $L'_i \leq C_i \leq U'_i$) does not contribute in the computation of LB'. The problem of uniform sampling is that, as it selects elements without following any particular criterion, it tends to select many elements from U and L that result to this third case. Therefore, LB' may become a significantly bad underestimation of LB that would have been computed if sampling was not used. The underestimation of the lower-bound value will result to an increase in false-alarms, which will incur high traffic.

To overcome this problem, we propose an alternative sampling method. We sample U and L separately. Initially, we store the elements of U in ascending order. In U' we select the first M elements of this ordering. Respectively, we sort L in descending order and we select the first M elements in L'. The intuition is that the selection of the smallest M values of U, helps in increasing the number of occurrences of the first case (i.e., when $C'_i > U'_i$), since the smallest the value of U'_i is, the more expected is to have a C'_i larger than it. Putting it less formally, we want to focus the sampling towards the 'gray' areas (see Figure 3 in Section 3). An analogous reasoning holds for the sampling of L'.

It is easy to see the following:

Lemma 1 *The sampling of U and L does not produce any false negatives.*

Proof. While computing LB', due to sampling, the first and second cases of Equation 4 occur less times than while computing LB (i.e., without sampling). Therefore, $LB' \leq LB$. Since $LB \leq D$ (where D is the actual distance, computed with DTW), we have that $LB' \leq D$. Thus, no false negatives are produced. □

The separate sampling of U and L presents the requirement of having to store the positions from which elements are being selected in U' and L'. If the positions are stored explicitly, then this doubles the amount of information kept ($2M$ numbers for storing U' and L' and additional $2M$ numbers for storing the positions of selected elements). Since this information is propagated during querying, traffic is increased. For this reason we propose an alternative representation. To represent U', we use a bitmap of length N (the phrase length). Each bit corresponds to an element in U. If an element is selected in the sample U', then its bit is set to 1, otherwise it is set to 0. Therefore, the combination of the bitmap and the M values that are selected in U' are used to represent U'. The same is applied for L'. This representation is efficient: the space required for U' is $M + \lceil N/8 \rceil$ bytes.[4] The plain representation requires $5M$ bytes (since it requires only one integer, i.e., 4 bytes, to store the position of each selected element). Thus, the proposed method is advantageous when $N < 32M$, i.e., for sample larger than about 3% (our experiments show that samples with size 10% are the best choice).

5 Similarity searching algorithms

In this section we present how existing algorithms for searching in P2P networks can be used within the proposed framework. The existing algorithms have been mainly used for searching text documents that are similar to a query text. Therefore, their straightforward implementation for searching similar music data does consider the significantly larger length of such data, and thus they are bound to incur excessive traffic. For this reason, we examine their incorporation in the proposed framework. More specifically, we focus on BFS, >RES, and ISM.

5.1 The BFSS algorithm

As previously explained, the simplest similarity searching algorithm is on the basis of breadth-first-search over the nodes of the P2P network. The adapted algorithm, which uses the proposed sampling and representation methods, is denoted as BFSS (breadth-first-search with sampling).

[4]Each element in an acoustic sequence is in the range 0-255, thus it requires one byte.

The pseudo-code for BFSS is given in Figure 4. Each time, the current node n is considered. A TTL value denotes how many valid hops are remaining for n, whereas T_s is the user-defined similarity threshold. It is assumed that sequences U' and L' carry also the associated bitmaps.

```
Procedure BFSS(Node $n$, int TTL, Sequence $U'$, Sequence $L'$, float $T_s$)
begin
1.  foreach data sequence $D$ in $n$
2.      foreach phrase $C$ of $D$
3.          $l = LB'(C, U', L')$
4.          if $l < T_s$
5.              get query sequence
6.              compute actual DTW distance, $D$, between phrase $C$ and query sequence
7.              if $D \leq T_s$
8.                  include $C$ in answer set
9.  if TTL $> 0$
10.     foreach peer $p$ of $n$ that has not been visited yet
11.         BFSS($p$, TTL-1, $U'$, $L'$, $T_s$)
end
```

Figure 4: The BFSS algorithm.

Evidently, the movement of the actual query sequence from the node that commenced the query to the currently visited node, increases the traffic (not being sampled, the query sequence has rather large length). For this reason, it is important not to have a large number of false-alarms.

The algorithm that does not use sampling (denoted as BFS) may produce less false-alarms. However, between each pair of peers it has to propagate U and L sequences, with length equal to the one of the query sequence. Therefore, it is clear that there is a trade-off between the number of additional false-alarms produced due to sampling and the gains in traffic from propagating sampled (i.e., smaller) envelopes. This trade-off is examined through the experimental results in the following section.

5.2 The >RESS algorithm

The >RES algorithm tries to reduce the number of paths that are pursued during searching. Instead of selecting, at random, a subset of the peers of the currently visited node, it maintains a profile for each such peer and bases its decision on this profile. In particular, each node maintains for each of its pears the number of positive answers that it has replied. Then, it selects the k peers that provided the most answers during the previous m queries. Both k and m are user specified.

It is clear that >RES algorithm can be easily adapted in the proposed framework. The query sequence will be sampled and represented according to the proposed method. This does not affect the profile that is maintained by >RES. The resulting method is denoted as >RESS (>RES with sampling). The pseudo-code for >RESS is given in Figure 5.

Since only a subset of peers is actually visited, >RESS tries to reduce traffic without missing a large number of answers. However, compared to BFSS, >RESS is expected to produce fewer answers. This tradeoff is examined in the next section, which contains the experimental results.

5.3 The ISMS algorithm

The ISM algorithm shares the same objective with >RES, i.e., it tries to reduce the number of examined paths. However, the profile maintained for each peer is different. ISM does not

```
Procedure >RESS(Node $n$, int TTL, Sequence $U'$, Sequence $L'$, float $T_s$, int $k$, int $m$)
begin
1.  foreach data sequence $D$ in $n$
2.      foreach phrase $C$ of $D$
3.          $l = LB'(C, U', L')$
4.          if $l < T_s$
5.              get query sequence
6.              compute actual DTW distance, $D$, between phrase $C$ and query sequence
7.              if $D \leq T_s$
8.                  include $C$ in answer set
9.  if TTL $> 0$
11.     $P_k$ = set of $k$ peers that provided most answers for the $m$ previous queries
12.     foreach peer $p \in P_k$
13.         >RESS($p$, TTL-1, $U'$, $L'$, $T_s$, $k$, $m$)
14.         Update the profile of $p$
end
```

Figure 5: The >RESS algorithm.

base its decision only on the number of answers to previous queries, but it also examines the similarity between the previously answered queries and the current one. Therefore, for each peer, a node maintains the t most recent queries that were answered by the peer. When a new query q arrives in the node, then it computes the similarity Qsim between q and all queries that are maintained in the profile of each node. A relative-ranking measure is given to each peer P_i, using the following formula:

$$RR_Q(P_i, q) = \text{Qsim}(q_j, q)^{\alpha} \times S(P_i, q_j)$$

where S(P$_i$,q_j) is the number of the results returned by P$_i$ for query q_j. Thus, ISM ranks higher the neighboring peers that return more results by adjustment of the α parameter. To make comparison more clear, we set α equal to 1, therefore we focused only on the criterion of similarity. We also have to notice that ISM may become biased towards the nodes that have answered somewhat similar queries in the past and may not give the chance to new nodes to be explored. For this reason, the following heuristic is used in [7]: besides the peers selected with the aforementioned criterion, ISM also selects at random an additional very small subset of peers (e.g., one node). In total, k peers are selected, where k is user-defined. The length of each profile (the number of queries stored in it) is also user-defined.

In order to adapt ISM to the proposed framework, we have to consider how to maintain the previously answered queries. In the proposed framework, query sequences are represented by their samples. Therefore, we measure the similarity between the current query's sample and the samples of previously answered queries. For this reason we maintain the samples of the answered queries in the profiles of the peers. To save time during the computation of the ranking, instead of measuring the actual similarity (through the DTW measure), we compute the *LB_Keogh* value. The resulting algorithm is denoted as ISMS (ISM with sampling). The pseudo-code for ISMS is given in Figure 6.

ISMS is expected to have slightly larger traffic than >RES, since it propagates the sample of an answered queries to all nodes involved in the search (in order to update their profiles). However, by testing the content of the queries, it tries to reduce the number of missed answers.

```
Procedure ISMS(Node $n$, int TTL, Sequence $U'$, Sequence $L'$, float $T_s$, int $k$)
begin
1.  foreach data sequence $D$ in $n$
2.      foreach phrase $C$ of $D$
3.          $l = LB'(C, U', L')$
4.          if $l < T_s$
5.              get query sequence
6.              compute actual DTW distance, $D$, between phrase $C$ and query sequence
7.              if $D \leq T_s$
8.                  include $C$ in answer set
9.  if TTL $> 0$
11.     $P_k$ = set of $k$ peers that have the most similar queries in their profiles
12.     foreach peer $p \in P_k$
13.         ISMS($p$, TTL-1, $U'$, $L'$, $T_s$, $k$)
14.         Update the profile of $p$
end
```

Figure 6: The ISMS algorithm.

6 Experimental results

The performance of the considered similarity searching algorithms was compared through simulation. The P2P network had 100 nodes and the average number of neighbors for each node was a random variable with average value equal to 7 (this kind of topology is called logarithmic). We used 500 real acoustic sequences, which correspond to various pop songs. Each song was sampled at 11 KHz and the average duration was about 4 minutes. To represent the fact that music songs (especially popular ones) are shared among several nodes, we replicated each sequence. The number of replications for each sequence was randomly variable with average value equal to 10.

The experimental comparison has two objectives: (i) The first objective is to determine the efficiency of the proposed framework. For this reason, we compare the proposed sampling and representation methods against the approach that does not use sampling and against a simplistic approach that uses uniform sampling. For clarity of comparison, we use the basic similarity searching method, i.e., BFSS. (ii) The second objective is to compare the examined similarity searching algorithms (BFSS, >RESS, ISMS) within the proposed framework. The evaluation metric in all cases is the average traffic (measured in MB) that each query incurs. The parameters we examine are: the sample size, query size (length of query sequence), query range (the user-defined threshold for similarity), and TTL value (max allowed number of hops). For the comparison of the examined algorithms within the proposed framework, we additionally examine the number of found matches (denoted as relative recall).

We start by focusing on objective (i) and compare BFSS against uniform sampling (this method is denoted as BFSS-UNI). The results are depicted in Figure 7. Figure 7a illustrates the relative traffic between BFSS and BFSS-UNI (i.e., the traffic of the latter is normalized w.r.t. the traffic of the former) against the query range. As shown, BFSS-UNI incurs about twice the traffic that BFSS does. As already explained, this is due to the fact that uniform sampling produces a bad underestimation of the lower bound value. This can be further understood when examining the discrepancy, denoted as error, between the bounds produced by BFSS and BFSS-UNI, and the actual bound produced by LB_Keogh. The relative error between BFSS and BFSS-UNI (i.e., the latter is normalized w.r.t. the former) is given in Figure 7b, against the query size. The error of BFSS-UNI ranges between 1.3 times the error of BFSS (for smaller queries) and 2.8 times (for medium sized queries). Therefore, the sampling method that is used in the proposed framework, performs better than uniform sampling.

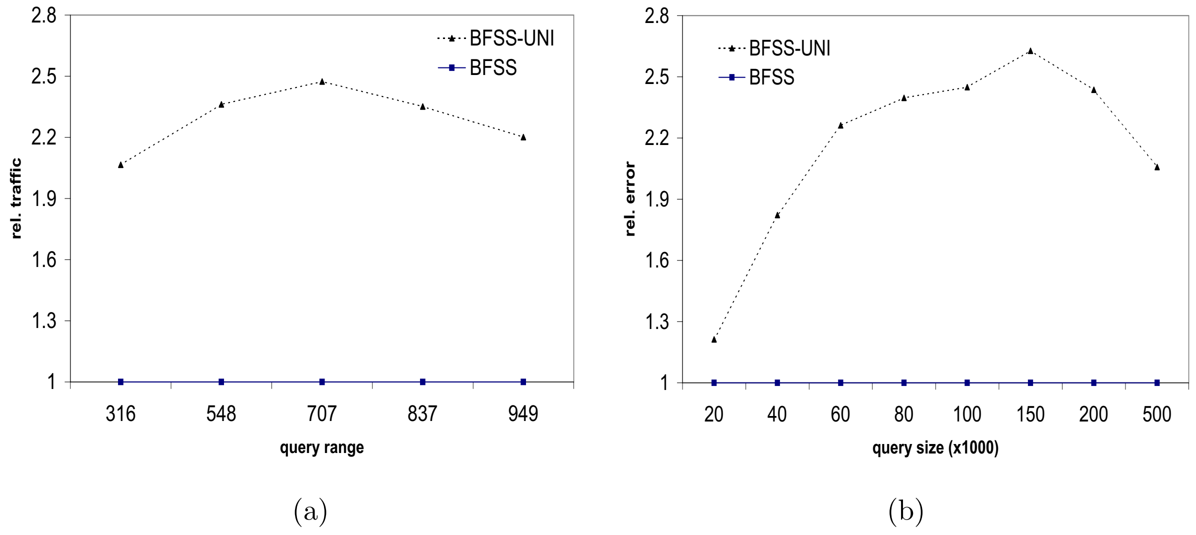


Figure 7: BFSS vs. BFSS-UNI (a) Relative traffic. (b) Relative error w.r.t. actual LB_Keogh value.

We now move on to compare BFSS against BFS (i.e., the method that does not use any sampling at all). For BFSS we examined several sample sizes. The results are depicted in Figure 8, whereas BFS has a constant value, as it does not use sampling. In Figure 8a, TTL was set to 4, query size was 100,000, and query range was set to 0 (i.e., exact match). As shown, for very small samples (with 1,000 elements), BFS performs better. This is expected, since the use of a very small sample affects BFSS by resulting to a large number of false alarms (due to bad underestimation of lower bound values), which increases traffic. However, by increasing the sample size, BFSS becomes better and clearly outperforms BFS. It is interesting to notice that the best performance is for sample size equal to 10,000 (i.e., 10% of the original query size). Finally, for large sample sizes, both methods converge to the same traffic. Analogous results are obtained for the case where TTL is set to 5. It also worths noticing that the traffic of BFS is significantly more increased than the traffic of BFSS does, compared to the case when TTL was 4. Therefore, the sampling that is used in the proposed framework helps in significantly reducing the traffic over the P2P network.

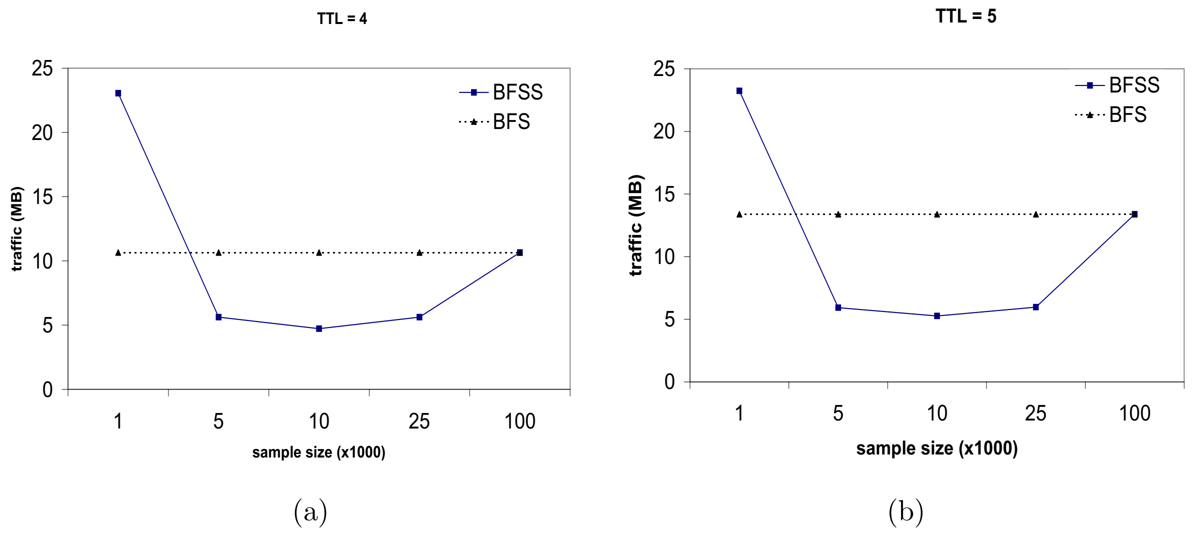
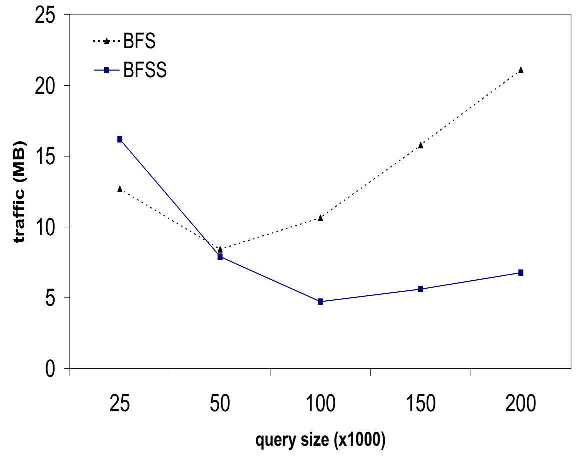
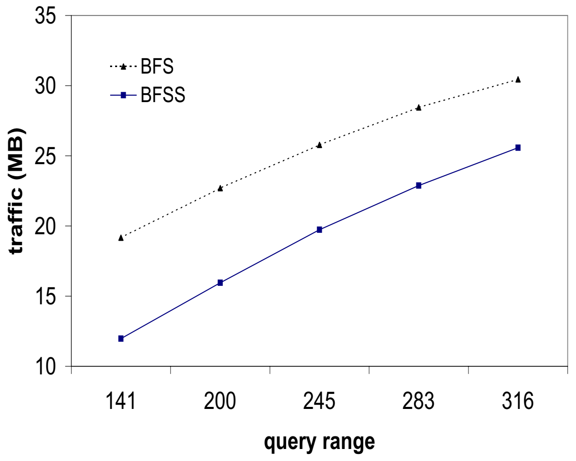


Figure 8: BFSS vs. BFS (a) Traffic (in MB) when TTL=4. (b) Traffic (in MB) when TTL=5.

Next, we compared BFSS against BFS for varying query size and query range, in order to test the sensitivity of the proposed framework. Figure 9a illustrates the results for the former case. The size of sample for BFSS was set each time to 10% of query size, TTL was set to 4 and query range was set to 0. As shown, BFSS clearly outperforms BFS in all cases, except for rather small queries. Again, for very small queries, the resulting sample is very small and many false-alarms are produced. Moreover, Figure 9b depicts the results for the latter case (varying query range). Query size was set to 100,000, sample size was 25%, and TTL was set to 5. BFSS clearly compares favorably with BFS.



(a)



(b)

Figure 9: BFSS vs. BFS w.r.t. (a) query size, (b) query range.

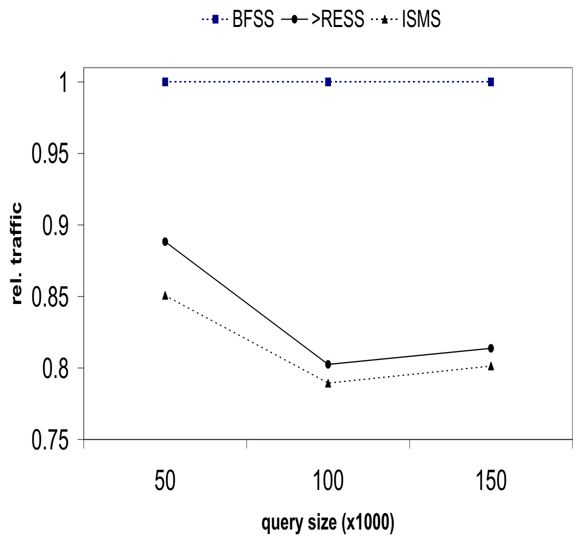
Finally, we examined objective (ii), i.e., the comparison of similarity searching algorithms within the proposed framework. We measured both the traffic and the recall achieved by BFSS, >RESS, and ISMS with respect to query size. TTL was set to 4 and query range was set to 0. The results on relative traffic (normalized by the traffic of BFSS) are presented in Figure 10a. As shown, both >RESS and ISMS require less traffic than BFSS. This is expected, since the former methods propagate queries only to a selected subset of peers. ISMS poses a little smaller traffic than >RESS, because it attains smaller recall (since Figure 10b). The results on relative recall (normalized by the recall of BFSS) are illustrated in Figure 10b. As expected, BFSS has the highest recall among all algorithms, since it visits all the peers of each node. >RESS comes second best, whereas ISMS has slightly worse recall. This is because we set the same maximum number of visited peers for >RESS and ISMS. By increasing this number for ISMS, the recall of both algorithms will be about the same, but the traffic of ISMS will also be increased in this case. We have to notice, however, that the recall of all algorithms is significantly high, since it is higher than 90%, which suffices for the application of retrieving similar music pieces.

7 Conclusions

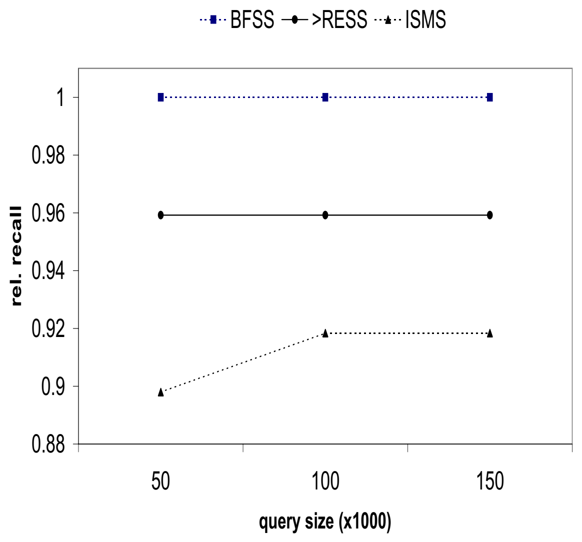
We have presented a novel framework for efficient retrieval of similar audio data. The proposed framework takes advantage of the absence of overhead in unstructured P2P networks and minimizes the required traffic for all operations with the use of an intelligent sampling scheme. Additionally, the framework has such a design that no false negative results occur.

Within the proposed framework we have adapted existing similarity searching algorithms, which have been previously proposed for finding similar text objects in P2P networks. Since their straightforward implementation is bound to result to significant traffic, their adaption in the proposed framework is necessary. For this reason, we showed how to modify them accordingly.

Detailed comparative evaluation illustrated the performance gains due to the proposed framework. We also analyzed the relative performance of the various similarity searching algo-



(a)



(b)

Figure 10: Comparison between similarity searching algorithms w.r.t. query size. (a) Relative traffic. (b) Relative recall.

rithms in the framework.

Future work includes the examination of other types of P2P networks (e.g., structured) and of additional similarity metrics.

References

[1] N. H. Adams, M. A. Bartsch, J. B. Shifrin and G. H. Wakefield: "Time Series Alignment for Music Information Retrieval", *Procceedings of the 5th Annual International Symposium on Music Information Retrieval*, 2004.

[2] R. Agrawal, C. Faloutsos and A. Swami: "Efficient Similarity Search In Sequence Databases", *Proceedings of FODO*, pp.69-84, Evanston, Illinois, USA, 1993.

[3] R. Agrawal, K.-I. Lin, H. S. Sawhney and K. Swim: "Fast Similarity Search in the Presence of Noise, Scaling, and Translation in Time-Series Databases", *Proceedings of VLDB*, Zurich, Switzerland, 1995.

[4] K. Chan and A. W. Fu: "Efficient Time Series Matching by Wavelets", *Proceedings of IEEE International Conference on Data Engineering (ICDE)*, pp.126-133, 1999.

[5] C. Faloutsos, M. Ranganathan and Y. Manolopoulos: "Fast Subsequence Matching in Time-Series Databases", *Proceedings of ACM SIGMOD*, pp.419-429, Minneapolis, Minnesota, USA, 1994.

[6] J,-S. R. Jang, H.-R. Lee and J.-C. Chen: "Super MBox: An Efficient/Effective Content-based Music Retrieval System". *Procceedings of the ninth ACM international conference on Multimedia*, pp.636-637, 2001.

[7] V. Kalogeraki, D. Gunopulos and D. Zeinalipour-Yazti: "A Local Search Mechanism for Peer-to-Peer Networks". *Procceedings of CIKM*, pp.300-307, 2002.

[8] E. Keogh: "Exact Indexing of Dynamic Time Warping". *Procceedings of VLDB*, pp.406-417, 2002.

[9] E. Keogh and A. N. Ratanamahatana: "Exact Indexing of Dynamic Time Warping", *Knowledge and Information Systems*, 2004.

[10] A. Klapuri: "Automatic music transcription as we know it today". *Journal of New Music Research*, to appeer, 2004.

[11] E.W. Large and C. Palmer: "Perceiving temporal regularity in music". *Cognitive Science*, Vol. 26, No.1, pp.1-37, 2002.

[12] M. Kontaki and A. N. Papadopoulos: "Similarity Search in Streaming Time Sequences", *Proceedings of SSDBM*, Santorini, Greece, 2004.

[13] X. Li and J. Wu: "Searching Techniques in Peer-to-Peer networks", *accepted to appeer in Handbook of Theoretical and Algorithmic Aspects of Ad Hoc, Sensor, and Peer-to-Peer Networks*, CRC Press, 2004.

[14] D. Mazzoni and R. B. Danneberg: "Melody matching directly from audio", *Procceedings of the Second Annual International Symposium on Music Information Retrieval*, pp. 17-18, 2001.

[15] A. Nanopoulos, Y. Theodoridis and Y. Manolopoulos: "C^2P - Clustering with Closest Pairs", *Proceedings 27th International Conference on Very Large Data Bases (VLDB'01)*, pp.331-340, Roma, Italy, 2001.

[16] P. Shrestha and T. Kalker: "Audio Fingerprinting in Peer-to-peer Networks", *to appear in Procceedings of the 5th International Symposium on Music Information Retrieval*, 2004.

[17] G. Tzanetakis, J. Gao and P. Steenkiste: "A Scalable Peer-to-Peer System for Music Information Retrieval", *Computer Music Journal*, Vol. 28, No. 2, pp.24-33, 2004.

[18] C. Wang, J. Li and S. Shi: "A Kind of Content-Based Music Information Retrieval Method in a Peer-to-Peer Environment", *Procceedings of the International Symposium on Music Information Retrieval*, pp.178-186, 2002.

[19] J.-Y. Won,J.-H. Lee, K. Ku, J. Park and Y.-S. Kim: "A Content-Based Music Retrieval System Using Representative Melody Index from Music Databases", *To appeer in ADBIS 2004*, 2004.

[20] B. Yang and H. Garcial-Molina: "Improving Search in Peer-to-Peer Networks". *Procceedings 22nd Inernational Conference of Distributed Computer Systems*, pp.5-15, 2002.

[21] C. Yang: "Efficient Acoustic Index for Music Retrieval with Various Degrees of Similarity". *ACM Multimedia*, pp.584-591, 2002.

[22] C. Yang: "Peer-to-peer architecture for content-based music retrieval on acoustic data". *Procceedings of the 12th International Conference on World Wide Web*, pp.376-383, 2003.

[23] B.-K. Yi and C. Faloutsos: "Fast Time Sequence Indexing for Arbitrary Lp Norms", *Proceedings of VLDB*, Cairo, Egypt, 2000.

[24] B.-K. Yi, H. V. Jagadish and C. Faloutsos: "Efficient Retrieval of Similar Time Sequences Under Time Wraping", *Proceedings of IEEE International Conference on Data Engineering (ICDE)*, pp.201-208, Orlando, Florida, 1998.

[25] Y. Zhu and D. Shasha: "Warping Indexes with Envelope Transforms for Query by Humming". *Procceedings of the 2003 ACM SIGMOD international conference on Management of data*, pp.181-192, 2003.