Multimed Tools Appl (2007) 32:49–71
DOI 10.1007/s11042-006-0068-5

Finding maximum-length repeating patterns in music databases

Ioannis Karydis · Alexandros Nanopoulos · Yannis Manolopoulos

Published online: 25 October 2006

Abstract This paper introduces the problem of discovering maximum-length repeating patterns in music objects. A novel algorithm is presented for the extraction of this kind of patterns from a melody music object. The proposed algorithm discovers all maximum-length repeating patterns using an "aggressive" accession during searching, by avoiding costly repetition frequency calculation and by examining as few as possible repeating patterns in order to reach the maximum-length repeating pattern(s). Detailed experimental results illustrate the significant performance gains due to the proposed algorithm, compared to an existing baseline algorithm.

Keywords Maximum-length repeating patterns · Data mining · Theme discovery · Music databases

1 Introduction

The continuously increasing spread of music on the Internet as well as in digital music libraries expands the already immense interest of the public and the entertainment industry in music databases. An account of research and development issues concerning a variety of digital music libraries can be found in [4]. As the number of music databases grows rapidly, so does their size, complexity, usage and accordingly the need to provide flexible and efficient search and retrieval techniques. Music data, due to their complex structure and their subjectivity to inaccuracies caused by perceptual

I. Karydis · A. Nanopoulos · Y. Manolopoulos (✉)
Department of Informatics, Aristotle University, Thessaloniki 54124, Greece
e-mail: manolopo@csd.auth.gr

I. Karydis
e-mail: karydis@delab.csd.auth.gr

A. Nanopoulos
e-mail: alex@delab.csd.auth.gr

and cognitive effects, introduce new challenges. Among the primary problems that have been examined so far, lays the development of representation types that can satisfy both semantic as well as efficiency requirements (since music objects exist in different formats) for content-based information retrieval.

A characteristic representation type for music objects is based on the use of *repeating patterns* included in a music object, i.e., segments of the music object that appear repeatedly (cf., Section 3). In this type, a repeating pattern corresponds to a *motif*, that is, a minimum-length pattern which is meaningfully independent and complete within a piece of music. Repeating patterns can constitute a useful representation for a music object. Their use (through the notion of motifs) has been extensive through out the history of music [5] as well as in modern music [3], since they comprise a compact form for indexing the original formats (e.g., raw audio, MIDI, etc.). This is because the total size of the collection of repeating patterns is smaller than that of the music objects. Therefore, the repeating patterns meet both efficient and semantic requirements for content-based music data retrieval [13, 19, 20]. For the aforementioned reasons, repeating patterns have been used to index music sequences for the purposes of music data retrieval [20]. In addition, they provide a reference point for the discovery of music *themes* [28, 39]. A theme (especially in classical music) is a melody that the composer uses as a starting point for development, which may be repeated in the form of variations[1] [39]. Finally, repeating patterns have been considered as *characteristic signatures* of music objects, which have the notion of a quantitative measure for music similarity [14].

For the problem of efficient discovery of repeating patterns, recent research has employed data mining techniques [20, 25, 28, 39]. As the straightforward use of repeating patterns may conceal numerous difficulties, mainly due to their large number, focus has been given on *non-trivial repeating patterns* [20, 28]. Nevertheless, the number of non-trivial patterns can still be large enough so as to burden their examination by human analysts. For instance, music objects with size about 1,000 notes can include several tens non-trivial repeating patterns [20], whereas this number increases for larger musical pieces. This can also impact the effectiveness of non-trivial repeating patterns in yielding themes, as several of the former may be spurious and unrelated to themes. Thus, existing research has identified that among the collection of the non-trivial repeating patterns, the longest ones are those that can be characterized as *feature melody strings* and are typically those that can yield to themes [28]. This is further analyzed in [20], where it is indicated that the longest repeating patterns (constrained by a maximum length value, e.g., 30) are most likely those that themes are based upon. Following this direction, [39] proposes a theme discovery method that is based on an initially computed collection of the longest repeating patterns.[2]

A straightforward approach for the discovery of the longest repeating patterns would include their selection in a post-processing step, following the mining of

[1]The variation extent and the repetition frequency of a theme can differ depending on the composer and the type of music (e.g., classic versus popular music).

[2]It is worth noticing, that the longest patterns discovered should be examined against several characteristics (e.g., frequency, duration, rhythmic consistency, position) [29] in order to effectively lead to theme discovery. Nevertheless, similar to [20], our focus is on the process of identifying the (longest) repeating patterns. Thus the examination of such factors is out of the scope of this paper.

all (non-trivial) repeating patterns. However, the length of the longest repeating patterns usually tends to be large (experiments in [20, 25] show that it can reach several tens). Therefore, the straightforward approach becomes rather inefficient, as a large number of intermediate repeating patterns (i.e., that are not the longest) have to be examined before reaching the longest ones. What is, therefore, required is the development of new algorithms that efficiently discover the longest repeating patterns, while not having to undergo the discovery of many intermediate repeating patterns.

As the number of these patterns can be up to several tens, efficiency issues require the avoidance of costly mining calculations by examining as few as possible intermediate patterns in order to reach fast the set of maximum-length repeating patterns (MLRPs). Interestingly enough, analogous reasoning has been followed in other data mining fields, e.g., in the mining of the longest itemsets [7, 27, 43]. Nevertheless, there are important differences (extensively discussed in Section 2.3) between the latter problem and the mining of the longest repeating patterns. Briefly, the key points of these differences are that approaches for long itemsets mining focus on large, disk-resident itemset databases while for the discovery of the longest repeating patterns, music sequences are main-memory resident and algorithms prioritise improved CPU times. In addition, in the problem of finding MLRPs, algorithms have a repeating pattern frequency threshold equal to one, while in mining itemsets algorithms such a consideration would produce a large overhead.

Finally, any proposed algorithm should take care of the characteristics that result from the nature of the examined problem, such as the ordering of notes or their replication within music sequence, factors that do not appear in related fields works like the frequent itemsets mining.

1.1 Contribution and outline

In this paper, we examine the problem of finding the maximum-length repeating patterns in music databases. Based on prior work on repeating patterns, we focus on note-sequences. The use of note-sequences for the extraction of repeating patterns has also been adopted by several previous works, e.g., [20, 25], since it is by note-sequences that music is composed and music semantics are conveyed to listeners. We propose a novel algorithm that discovers all maximum-length repeating patterns using a fast ascending, as far as the length of the patterns is concerned, during searching so as to quickly reach these patterns. To achieve this, the proposed algorithm avoids the examination of a large number of intermediate patterns (i.e., of not maximum length) and only considers those patterns that are necessary in order to reach the maximum-length pattern(s).

The technical contributions of this paper are summarized as follows:

- The introduction of the problem of discovering maximum-length patterns in music objects. In the field of music databases, this problem poses significant requirements due to the very large length such patterns may have (i.e., a large search space).
- The development of a novel algorithm that efficiently discovers the maximum-length patterns. The proposed algorithm addresses the characteristics that result from the nature of the examined problem, i.e., factors like the ordering of notes

or their replication within music sequence (such factors do not appear in work in related fields like the frequent itemsets mining).
- The detailed experimental results which show the efficiency of the proposed algorithm, and the performance gains compared to an existing baseline algorithm [20].

The rest of the paper is organized as follows. Section 2 is devoted in related research in terms of music databases, music pattern discovery, data mining and, in particular, in long patterns and their unsuitability to be directly applicable. Section 3 discusses one of the most efficient approaches already researched and describes the motivating issues for this research. Extending the idea proposed in Section 3, Section 4 provides a complete account of the algorithm proposed in this paper (supported by a running example). Subsequently, Section 5 presents and discusses the experimentation and the results obtained. Finally, the paper is concluded by a summary and possible future work in Section 6.

2 Related work

2.1 Work in music databases

Early works on music information retrieval date back on 1966 [24]. Despite this—rather ahead of its time—work, very little was done for many years thereafter, until lately an explosive interest on music IR arose. Many disciplines of music IR have been researched including the types of queries allowable, similarity algorithms, various mapping schemes for the music objects and a range of indexing techniques.

Based on the well-studied text, image and video data IR, music IR can be performed using text (metadata) [2], pieces of structured or unstructured music [6, 16, 23, 41], humming [6, 10, 15, 17, 31] or even classic western musical notation [32] as queries.

Music is available in three basic representations: Audio, Time-stamped Events and Common Music Notation [8]. To achieve semantic, efficiency as well as to overcome data processing constraints, the previously mentioned basic representations require a mapping. The mapping process is influenced by music perception and psychoacoustic issues. That is, a variety of alternatives exist with respect to the characteristic of music that should be retained by the mapping. In [11, 36] the retained characteristic of music is rhythm and the mapping used leads to rhythm strings. In [10] music melody and contour are mapped.Chen H. and Chen A.L.P. [12] focuses on properties such as pitch, duration and loudness, while [16, 23, 41] retain the melody of the music object.

In the field of time-series analysis, several methods have been proposed for various tasks, e.g., regression, classification, and similarity searching [18]. Nevertheless, the particularities of music sequences and the different application requirements have lead to the development of novel methods. For instance, the similarity of two mapped music objects is addressed in numerous approaches in the literature [17, 29, 34, 40], which mainly depend on the string matching core technique since, most usually, the mapping procedure for a music object produces a string of a chosen characteristic.

A number of works [35, 38, 42] have utilised Hidden Markov Models (HMMs) in order to represent music pieces in a database and the queries posed. In [35], Pikrakis,

et al. present a method for automated search of predefined sound patterns within a large number of sound files, using HMMs. In [38] the authors use a stochastic representation of both music sequences in the system and the queries, with hidden Markov models in order to handle queries that contain errors or key and tempo changes. Velivelli et al. [42] utilise HMMs that can model predefined patterns and simultaneously identify and match an audio segment for a given query. All the aforementioned works have utilised HMMs for the purposes of musical query retrieval, whereas the focus in the research field of theme discovery [20, 25] is to find patterns within the music sequences.[3]

Managing the extracted features from the original music objects requires the utilization of an index scheme in order to keep the retrieval time close to constant. A number of alternatives have been proposed by [9, 11, 17] in order to manage different features extracted from music data and support several search functions.

2.2 Mining repeating patterns and theme discovery

The process of mining repeating patterns is described in [20, 28], where two algorithms are proposed for the discovery of non-trivial repeating patterns and feature melody string. The first algorithm uses a correlative matrix for the extraction of repeating patterns, while the second is based on a repeating string-join operation. Experimental results in [20, 28] indicate the superiority of the latter algorithm in comparison to the correlative matrix approach. More details for the string-join-based method are given in Section 3.2 along with the motivation. Koh and Yu, [25] presented a means of mining the maximum repeating patterns from the melody of a music object using a bit index sequence as well as an extension for extraction of frequent note sequences from a set of music objects. In the approach taken in [25], all repeating patterns are found and verified by counting their frequency, while redundancy examination is performed as a latter step, reaching the maximal repeating pattern set rather inefficiently. Rolland and Ganascia, [37], propose an approach for approximate sequential pattern extraction in music data, which considers several peculiarities of music objects and is based on the definition of a similarity function.

As far as the use of repeating patterns in theme discovery is concerned, Smith and Medina [39] proposed a pattern matching technique leading to theme discovery, that is based on a collection of previously found longest repeating patterns. Meek and Birmingham in [29] identify numerous features, that need to be extracted from each music object for the discovery of themes. Among them, they considered as most important the position of the theme (favoring the themes appearing earlier in the music object). As described, such features can be used for the discovery of themes from the repeating patterns found. Thus, both [39] and [29] can be considered as complementary to the problem described in this paper. In addition, an interesting web-based system for theme discovery is presented in [26].

Patterns may not only be in one voice (the case of polyphonic music), as a pattern may be distributed across several simultaneously sounding voices. [21] and [22] present a number of different algorithms for the discovery of such patterns, including distributed pattern matching with at most k-differences (motif evolution).

[3] Regarding the related work on sequence mining (e.g., [1]), please read the corresponding description in Section 2.3.

The previously mentioned works primarily address the problem of finding all repeating patterns and generally are concerned with their relation with the set of themes. In contrast, our work focuses on finding all MLRPs,[4] a problem which, to our knowledge, has not been examined so far in the context of musical data. At this point the semantic value of MLRPs must be addressed. Existing results in [20, 28] indicate that "many repeating patterns (*at least for the longest ones*) of a real music object are intentionally created by its composer" [28]. Accordingly, the existence of the MLRPs (i.e., the longest patterns) is intentional. Therefore, the need for their discovery is evident, since it yields to information about the composer's objective.

Moreover, as previously mentioned, MLRPs are themselves repeating patterns (RPs) and additionally they "contain" all RPs that can be derived as their subsequences. Thus, MLRPs inherently carry the semantic value of the corresponding RPs (the MLRPs themselves and the RPs that are their subsequences). The semantic value of RPs is analyzed in [12, 20, 25, 28]. In particular, the experimental results in [28] illustrate an 100% recall in extracting musical motives from repeating patterns (i.e., a motive must always be a repeating feature). Results in [20, 28] show that the clustering of music objects can effectively be done based on repeating patterns. However, it should be made clear that MLRPs are patterns that are designated to reveal a different, new aspect of the music objects. Since we are not interested in finding all the existing repeating patterns, we do not focus on relating the proposed type of patterns with the set of all motives.[5]

2.3 Mining long itemsets

In the field of itemsets mining, several methods have been proposed recently for the discovery of the maximum-length frequent itemsets [7, 27, 43]. The focus of these methods is to avoid the examination of all frequent itemsets, moving the search towards the fast discovery of the itemsets having the maximum length or those being the maximal (i.e., that have no superset that is also frequent). Obviously, there is distinct analogy between the problem examined in [7, 27, 43] and the problem of discovering MLRPs. However, the process of mining MLRPs presents important differentiations due to which the aforementioned approaches cannot be directly applied.

To begin with, the principal difference of the approaches for itemsets is their focus on large, disk-resident itemset databases. Accordingly, the techniques involved in [7, 27, 43] reduce the number of database scans while utilizing structures that are optimized to address the volume of data. In contrast, for the problem of mining repeating patterns and MLRPs, the music sequence is main-memory resident, and the involved structures and techniques have the objective of performing fast operations to improve the CPU time. Therefore, the application of existing techniques for itemsets would be inefficient in this domain, since the optimizations they consider

[4] C.f., MLRPs are the repeating patterns with the maximum length—see Definition 3.

[5] This is the reason why we did not explore a systematic comparison of our results against manual references, e.g., [5], since it is out of the scope of the proposed work to relate with the identification of all motives. Nevertheless, we found that for several classic music objects the discovered MLRPs (or at least some of their subsequences) could identify some of the motives of the examined music objects.

mainly target the I/O cost. This is the reason why [20] did not consider the direct use of a mining technique for sequence databases, like [1], for the problem of mining repeating patterns in music sequences. Additionally, in the problem of mining repeating patterns and MLRPs, a subsequence of the music sequence is a pattern if its frequency is greater than 1. In contrast, the algorithms that mine itemsets consider a much larger threshold for the frequency of patterns[6] and, most importantly, they are bound to have a large overhead should they consider the frequency threshold to be equal to 1.

Based on the aforementioned works, we propose a novel method that targets the specific application requirements, i.e., we consider optimizations for main-memory resident music sequences and for patterns that can appear at minimum twice within the sequence (frequency threshold to be equal to 1).

3 Background and motivation

3.1 Definitions

We consider a music sequence to be a sequence of symbols from an alphabet containing discrete elements. In general, music is characterized by several features. Among them, pitch, rhythm, timbre, and dynamics are considered to be the most semantically important ones [8]. For western music in particular, the pitch carries the highest relative weight of information [8]. In general, rhythm cannot be overlooked, for easier presentation however, we focus on the pitch information. Notice that this assumption has been followed in all related work on the discovery of repeating patterns [20, 25, 39]. Nevertheless, it is easy to notice that the proposed methodology can be easily applied to rhythm sequences. A more challenging task is to combine both the important features (i.e., pitch and rhythm) in the discovered patterns. In this case, though, slight variations in the themes will produce different combined sequences. What is required, then, is the development of methods that will not be sensitive to small variations so as not to loose many repeating patterns. For the above reasons, this research direction will be addressed in future work.

Definition 1 (Repeating pattern [20]) Given a music sequence S, a repeating pattern P is a subsequence of consecutive elements of S that appears at least twice in S.

Notice that for the MIDI representation, the size of the alphabet (number of distinct elements) is equal to 128. The repeating frequency $freq(P)$ (hereafter called frequency) of a repeating pattern P is defined as the number of appearances of P in S. The length $|P|$ of a repeating pattern P is the number of notes in P.

Definition 2 (Maximal repeating pattern [25]) A repeating pattern X is a maximal repeating pattern in a music sequence S, if X is a repeating pattern in S and there does not exist another repeating pattern X' in S such that: (1) X is a subsequence of X', and (2) the $freq(X) = freq(X')$.

[6] Even small percentages of frequency thresholds, e.g., 0.1%, correspond to a much larger value than the absolute value of 1.

Definition 3 (Maximum length repeating pattern) A repeating pattern X is a maximum length repeating pattern in a music sequence S if: (1) X is a maximal repeating pattern of S, and (2) there does not exist another repeating pattern X' in M for which $|X'| > |X|$.

The above definition initially requires for a repeating pattern X, in order to be a maximum length repeating pattern, not to be a subsequence of another repeating pattern X', with which they have the same frequency, in which case X' would be the maximal. In addition the definition requires that X has the biggest length of any other repeating pattern X'. For example, in a sequence S = EABCDEBCABCDBCA, there exist 13 repeating patterns, of which $\{ABCD\}$ is the maximum-length repeating pattern (since it is maximal and there does not exist another repeating pattern X' in S for which $|X'| > |X|$) , $\{A, BC, BCD, BCA\}$ are maximal while the rest are trivial.

Finally, the definition of the problem examined in this paper is as follows: given a music sequence S, find all (if any) MLRPs.

3.2 The HLC algorithm—our baseline competitor

As already discussed in Section 2, Hsu et al. [20] proposed two different techniques for the discovery of non-trivial repeating patterns. Herein, we focus on the string-join approach, which is denoted as HLC (from the initials of the authors' names). HLC will be epigrammatically considered (through a concise example), so as to describe its suitability as a baseline algorithm for the extraction of MLRPs (cf. Section 3.3).

HLC utilizes $\{X, freq(X), (pos_1, pos_2, \ldots)\}$ to represent a repeating pattern found in a music sequence S, where X is the repeating pattern, $freq(X)$ is the repeating frequency of X and each $pos_i, 1 \leq i \leq freq(X)$, is a starting position of X in S. According to [20] the string-join operation is defined as follows: assume that $\{\alpha_1\alpha_2 \ldots \alpha_m, freq(\alpha_1\alpha_2 \ldots \alpha_m), (p_1, p_2, \ldots, p_i)\}$ and $\{\beta_1\beta_2 \ldots \beta_n, freq(\beta_1\beta_2 \ldots \beta_n), (q_1, q_2, \ldots, q_j)\}$ are two repeating patterns in the music feature string of a music object. We define order-k string-join ($k \geq 0$) of the two repeating patterns as follows:

$$\{\alpha_1\alpha_2 \ldots \alpha_m, freq(\alpha_1\alpha_2 \ldots \alpha_m), (p_1, p_2, \ldots, p_i)\} \bowtie_k \{\beta_1\beta_2 \ldots \beta_n, freq(\beta_1\beta_2 \ldots \beta_n), (q_1, q_2, \ldots, q_j)\} = \{\gamma_1\gamma_2 \ldots \gamma_l, freq(\gamma_1\gamma_2 \ldots \gamma_l), (o_1, o_2, \ldots, o_h)\}$$

where

- $i = freq(\alpha_1\alpha_2 \ldots \alpha_m)$, $j = freq(\beta_1\beta_2 \ldots \beta_n)$, $h = freq(\gamma_1\gamma_2 \ldots \gamma_l)$,
- $\gamma_t = \alpha_t$ for $1 \leq t \leq m$, $\gamma_t = \beta_{t-m+k}$ for $m+1 \leq t \leq l = m+n-k$,
- $o_t = x = y - m + k$, where $x \in \{p_1, p_2 \ldots, p_i\}$ and $y \in \{q_1, q_2 \ldots, q_j\}$,[7]
- $o_t < o_{t+1}$, for $1 \leq t \leq h-1$,
- if $k > 0$, $\alpha_{m-k+s} = \beta_s$, for $1 \leq s \leq k$.

HLC develops in two stages: in the first stage, repeating patterns of length 2^k (initially, $k = 0$) are found, while repeating patterns of length 2^{k+1} are then found by joining repeating patterns of length 2^k. The search, during the first stage, proceeds

[7] This condition refers to how the position of elements in sequence γ relates to the positions of appearance of the sequences α and β.

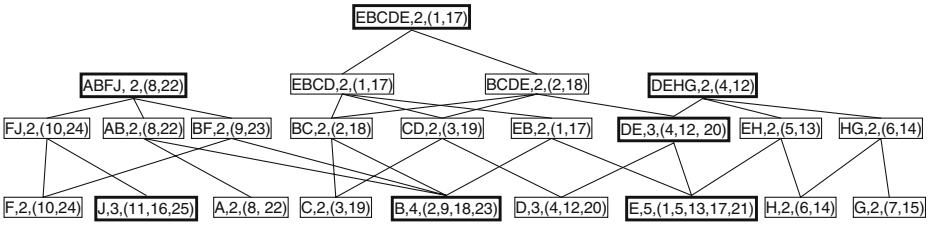


Fig. 1 The complete graph for the running example of the HLC

until a k_l is reached for which no repeating pattern exists. At this point, HLC has to determine the length L of the longest repeating pattern, which is unknown in advance. Though, the length of the maximum repeating pattern L is known to be between $2^{k_{l-1}} \leq L < 2^{k_l}$. Therefore, HLC performs a binary search of the patterns the length of which is in the range $[2^{k_{l-1}}, 2^{k_l})$. At the end of the first stage, the HLC has determined the L and the corresponding maximum-length patterns. Proceeding to the second stage, in order to ensure that all repeating patterns found in the previous step are non-trivial, a tree structure called RP-Tree is introduced, each node of which represents a repeating pattern found. After the removal of all trivial repeating patterns, a refining procedure identifies all repeating patterns the length of which is not a power of two (if any). The resulting repeating patterns of the refining process are added to the RP-Tree. Finally, all trivial repeating patterns are discarded, leaving the RP-Tree to contain only the maximum and the non-trivial repeating patterns, completing thus the second stage of HLC.

To clarify the understanding of HLC, we exemplify its execution over an example music sequence (that will be used as the running example throughout this paper). Let S be a music sequence, where S = EBCDEHGABFJDEHGJEBCDEABFJ. Following the previously discussion, the repeating patterns of length 1, 2, 4 do exist, though RP[8]=∅, where RP[x] denotes the set of repeating patterns with length equal to x. To determine L (and the corresponding maximum-length repeating patterns), we consider that $k_l = 3$, since $8 = 2^3$; whereas $k_{l-1} = 2$, since $4 = 2^2$ and RP[4] is the last length containing repeating patterns. Therefore, the algorithm searches the intermediate values of length 5, 6 and 7 discovering RP[5]={EBCDE,2,(1,8)} $RP[6] = \emptyset$ and $RP[7] = \emptyset$. Thus, $L = 5$ and the set of MLRPs is RP[5]={EBCDE,2,(1,8)} (i.e., RP[5] contains only one such pattern). The result of the first stage of HLC is illustrated in Fig. 1, where the MLRP is depicted at the root. (In Fig. 1, the non trivial repeating patterns are depicted with bold line.) The following stage of the HLC algorithm is of no interest to this research since it focuses on MLRPs (which are identified in the first stage), thus for brevity, the steps performed by the second stage of HLC are omitted.

3.3 Motivation

Based on the above discussion, it should be noted that, among the other non-trivial repeating patterns, HLC discovers the set of all MLRPs. Evidently, this is done in a quite efficient way, due to the following reasons: (1) Only a logarithmic number of intermediate lengths are considered to discover the MLRPs (lengths of the form of 2^k are examined until k_l is found and, then, a binary search is followed in the range

$[2^{k_{l-1}}, 2^{k_l})$), whereas a straightforward approach would check all possible lengths between 1 and L. (2) Through our experimental measurements we have found that the most time consuming stage of the HLC is the second stage, where the building of the RP-Tree and the elimination of the non-trivial patterns occur; in contrast, should the focus be only in finding the MLRPs (and not the set of all repeating patterns), then the second stage can be entirely omitted.

For the above mentioned reasons, a modified version of HLC (that consists only of its first stage) can be considered as a good baseline algorithm for comparison purposes, since it significantly outperforms the straightforward approach. Nevertheless, it must be mentioned that HLC was not designed to discover only the MLRPs. Although it approaches the set of MLRPs through a logarithmic number of intermediate levels, at each such examined level it has to identify *all* the repeating patterns of that level. As the maximum length can reach the order of several tens or even few hundreds, HLC has to join and count the frequency of a large number of repeating patterns. This is mostly evident during the initial steps, when the number of repeating patterns of relatively small lengths is very large, due to the anti-monotonicity property.[8] Therefore, a new approach is required that will avoid as much as possible the cost to examine (i.e., counting the frequency) intermediate patterns.

Finally, it should be taken into account that Koh and Yu [25] proposed a different approach for the discovery of repeating patterns. Their method utilizes a bit-index table and identifies all repeating patterns using a unit length increment. Therefore, the method [25] reaches the level of MLRPs by considering all intermediate lengths, and not a logarithmic number of levels as HLC does. Moreover, similarly to HLC, at each examined level, the method of [25] considers all repeating patterns. Experimental results in [25] indicate an improvement of the overall execution time compared to the HLC algorithm. Nevertheless, these results assumed the problem of finding all repeating patterns, where HLC had to undergo the expensive second stage. Therefore, the modified HLC is considered much more efficient than the method of [25], for the problem of discovering only the MLRPs. Thus, we select the (modified) HLC algorithm as the baseline method that we use for comparison purposes.

4 The proposed method

4.1 Outline of the approach

In this section we describe the proposed algorithm, which is denoted as M^{2}P (Mining Maximum-length Patterns). The outline of the approach taken by M^{2}P is as follows. Let $S = \langle s_1, \ldots, s_n \rangle$ be a music sequence of length n. Assume that we have identified all repeating patterns of length two, which is denoted as RP[2] $= \{\langle s_i, s_j \rangle : s_i, s_j \in S, freq(\langle s_i, s_j \rangle) \geq 2\}$. The elements of S and of RP[2] form a directed graph $G(V, E)$, where the set of vertices $V(G)$ corresponds to the set of all elements of S and the set

[8] According to the property of anti-monotonicity, a subsequence X of S cannot be a repeating pattern unless all the subsequences of X are also repeating patterns (we are not interested about the distinction between trivial and non-trivial patterns, since MLRPs are by definition non-trivial).

of all edges $E(G)$ to the set of all elements of RP[2] (i.e., a directed edge $\langle s_i \rightarrow s_j \rangle$ in the graph corresponds to the member $\langle s_i, s_j \rangle$ of RP[2]).

Each path P in G can be considered as a possible repeating pattern, since all its subpaths of length two (i.e., the directed edges) are repeating patterns. Therefore, the set of all possible paths of G forms the search space of the examined problem, as the MLRPs are also repeating patterns and, thus, correspond to paths of G. A naive approach would consider the complete graph, where each possible pair of elements of S would form an edge. However, this approach would lead to an excessive number of possible paths, whereas (due to the anti-monotonicity property) this number is drastically pruned, due to the fact that edges correspond only to members of RP[2].

The objective of M^{2}P is to identify in the aforesaid search space those paths that have maximum length and correspond to a repeating pattern. To attain this, M^{2}P traverses G by searching for the paths that originate from any of its vertices. While encountering paths, M^{2}P is concerned in identifying only these, which are candidates to become a MLRP (i.e., not only repeating patterns). During the traversal, it keeps track of the path C that has already been visited and: (1) has, so far, the maximum length, and (2) corresponds to a repeating pattern (i.e., its frequency has been counted and found to be larger than two).[9] The pruning of the search space is accomplished by discarding the extensions (i.e., appending of vertices and edges during the traversal) of paths that their frequency has been counted and they were not found to be repeating patterns, as none of their extensions can lead to an MLRP (due to anti-monotonicity, since an MLRP is a repeating pattern). Therefore, while advancing the traversal of G, three cases need be considered:

Case 1: if the currently visited path P has length smaller than $|C|$, then counting its frequency can be avoided (since it will definitely not be an MLRP).

Case 2: if $|P| > |C|$, then the frequency of the corresponding pattern in S is calculated, and if found to be a repeating one, then C is set to be equal to P. Otherwise, if not a repeating pattern, then (as already explained) the traversal does not have to follow any path containing P.

Case 3: finally, if P's length is equal to $|C|$, then the calculation of its frequency is avoided, at this point. Instead, we maintain a list and link it to C. If after the end of G's traversal no other repeating pattern has been found with length greater than $|C|$, all such paths linked to C are also candidates to be repeating patterns (C has been identified as an MLRP, because it was the first path of its length that was considered during the traversal, so its frequency has been counted due to case 1).

Following the previously discussed approach, M^{2}P calculates the frequency of a path only if its length is such that it can possibly become an MLRP. For this reason, it postpones as much as possible the costly operation of frequency calculation, aiming at finding new candidates with larger length. The result is that M^{2}P, unlike HLC, avoids calculating the frequency of all paths of a certain length. Instead, it only determines the frequency of paths of a given length, until the first path corresponding to a repeating pattern is found. Finally, when finished with the traversal, all candidates that are linked to the initially found MLRP (i.e., those with length equal to the found

[9] Initially, any edge of G can be selected as such path.

maximum found length for $|C|$) are examined so as to find all MLRPs, as there may be more than one. It should be noted that the frequency counting in M^{2}P is done by using a string matching algorithm,[10] since the frequency of a path P is equal to the number of appearances of P (i.e., of the sub-sequence corresponding to P) in S.

4.2 The M^{2}P algorithm

In this section we describe the algorithmic form of M^{2}P, which is depicted in Fig. 2. The input data of M^{2}P is the music sequence. Initially, M^{2}P calculates all repeating patterns of length 2 and stores them in the RP[2] set. This is done as an initialization step through a two dimensional array M, the size of which for the MIDI representation is 128×128. The graph G is constructed based on the adjacency matrix representation of M. Next, M^{2}P performs a traversal of G during which it examines the paths P originating from the vertices of G (the traversal visits the vertices in a depth-first manner).

Within the graph traversal procedure, the length of the current path P is compared against the Current Maximum Length path, which is denoted as CML (initially, it is set to 2, since M^{2}P has already determined the RP[2] set). If P's length is greater than CML, then M^{2}P counts the frequency of P and, in case it is greater than 2, P is stored (as the only element) in the Maximum Length Queue (denoted as MLQ), whereas CML is set equal to the length of P. In contrast, if P's length is equal to CML, then P is added to MLQ without counting its frequency. Finally, if the search for paths containing P has not been pruned (pruning occurs when P's frequency is counted and found less than two), the traversal continues further by visiting nodes adjacent to the last node v of P.

After the traversal of G has ended, M^{2}P has established (if any) one MLRP (the first element in MLQ). Therefore, it continues by calculating the frequency of all remaining members (if any) in the MLQ, to find the set of all MLRPs.

The correctness of M^{2}P can easily be deduced as follows. Assume that P_M is a MLRP whose length is M and its elements are $\langle p_1, \ldots, p_M \rangle$. Since P_M is a MLPR, its frequency is equal or greater than 2. Therefore, each consecutive pair $\langle p_i, p_{i+1} \rangle$ of P_M's elements belongs to RP[2] and has a corresponding edge in G. Accordingly, P_M will be examined by M^{2}P during the traversal of G, following the edges $\langle p_i, p_{i+1} \rangle$ for $1 \leq i < M$. If P_M is the first path with length M that is examined, then its frequency will be counted and P_M will constitute the first element of MLQ (by deleting any prior entries corresponding to candidates of smaller length). Otherwise, if other paths of length M have already been included in MLQ, since no other repeating pattern P' exists with $|P'| > M$, P_M will be examined in the step after the traversal has terminated, while counting the frequencies of all elements of MLQ. Thus, in either case, P_M will be included in MLQ and will be included to the output of M^{2}P.

4.3 Example

To clarify the description of M^{2}P, we give an example of its execution following the running example of the paper. For this example, $S = EBCDEHGABFJDEHGJE$

[10] For simplicity, in our implementation we used the Knuth–Morris–Pratt algorithm.

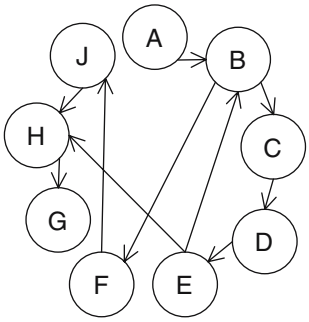
```
Procedure M²P(MusicSequence S)
begin
1.  RP[2] = Find all rp with length 2
2.  Construct G(RP[2])
3.  CML := 2
4.  MLQ := ∅
5.  for each v ∈ V(G)
6.        Traverse(G, v, ⟨v⟩, CML, MLQ)
7.  endfor
8.  for each P ∈ MLQ
9.        if (CountFreq(q)≥ 2)
10.             Output(P)
11.       endif
12. endfor
end

Procedure Traverse(Graph G, Vertex v, Path P, int CML, Queue MLQ)
begin
1.  bool prune := false
2.  Append(P, v)
3.  if Length(P) > CML
4.        if CountFreq(P ≥ 2)
5.             MLQ := P
6.             CML = Length(P)
7.        else
8.             prune := true
9.        endif
7.  else if length(P) = CML)
8.        Enqueue(MLQ, P)
9.  endif
10. if not prune
11.       for each u ∈ V(G) and ⟨v → u⟩ ∈ E(G)
12.            Traverse(G, u, P, CML, MLQ)
13.       endfor
14. endif
end
```

Fig. 2 The MLRP algorithm

BCDEABFJ, its RP[2] set and the corresponding graph G are illustrated in Fig. 3. Assume (without loss of generality) that the M²P begins its traversal from the paths emanating from vertex A and from edge AB in particular. Initially, path ABC is visited (Fig. 4a). Since its length is $3 > CML = 2$, its frequency is counted for and found equal to 0. Therefore, M²P does not continue the traversal following the path ABC. Then, it continues by examining ABF, whose frequency is counted equal to 2. Accordingly, CML is set to 3 and ABF is inserted in MLQ. The traversal continues further with this path, moving on to $ABFJ$, whose frequency is counted and found

Fig. 3 The example graph *G*



equal to 2. Similarly, *CML* is set to 4 and *MLQ*={ABFJ}. Furthermore, the path *ABFJH* is considered, but its frequency is counted to be equal to 0. Therefore, we avoid the examination of further paths that contain it.

Next, the traversal moves on to vertex *B* (Fig. 4b) and the edge *BC* in particular. To begin with, path *BCD* is examined, the length of which is less than *CML*, and thus its frequency is not counted. However, the traversal continues following paths containing *BCD*, since it cannot be discarded as not being a repeating pattern (i.e., we have not counted its frequency). Thus, path *BCDE* is next examined, whose length is equal to *CML*. Thus, *BCDE* is added to *MLQ* and *MLQ* becomes equal to {ABFJ,BDCE}.

Following a similar approach, paths emanating from vertex *C* (Fig. 4c) do no change *CML* or *MLQ*, while the paths resulting from vertex *D* (Fig. 4d) add *DEHG* to *MLQ* (since $|DEHG| = CML = 4$, its frequency is not calculated), while *MLQ* becomes equal to {ABFJ,BDCE,DEHG}. Moving on to vertex *E* (Fig. 4e), the path *EBCD* is added to the *MLQ* (*MLQ*={ABFJ,BDCE,DEHG,EBCD}). Next, path *EBCDE* is examined, and its frequency is counted (since its length is larger

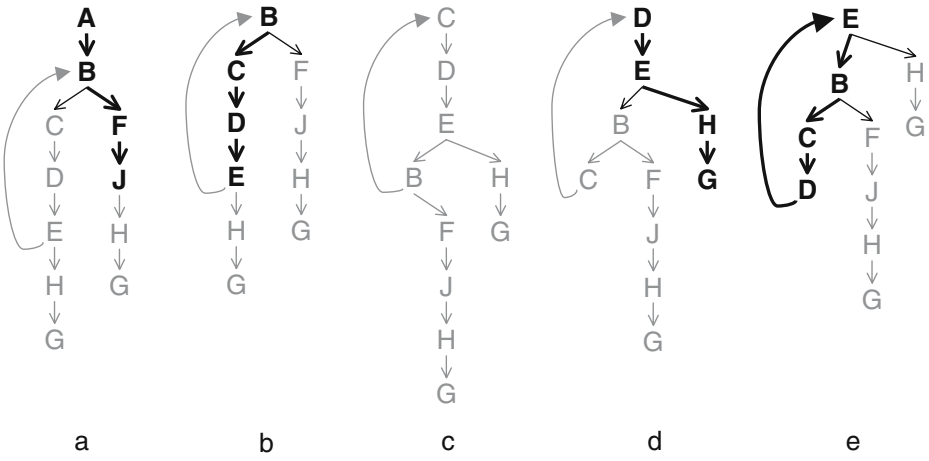


Fig. 4 Example of paths originating from vertices A, B, C, D, and E

than *MLQ*) and found equal to 2. Therefore, *CML* is set to 5, whereas the current elements of *MLQ* are removed and *EBCDE* is inserted in it (since a larger *CML* value has been found). Finally, all other vertices (*F*, *G*, *H* and *J*) offer no change. Thus, as no other candidates exist in the *MLQ* the set of found MLRPs is equal to {*EBCDE*}.

4.4 Developing optimizations

The efficiency of the M^{2}P algorithm rests with its two main features, the ability to avoid, as already described above, the calculation of the repeating frequency of the candidates (except for the first one found for each length) the length of which is equal to the CML, and the ability to avoid completely any measurement concerning candidates with length smaller than the CML. To improve further its efficiency, we describe two techniques that were used to enhance the basic form of M^{2}P.

As indicated in [20], the number of repeating patterns with small length is much higher than the number of repeating patterns with large length. For this reason, we would like M^{2}P (during the traversal) to reduce the number of examined paths with small length. This is attained in a preprocessing step. Let ℓ be the length of repeating patterns that we are interested in reducing their number. M^{2}P reads the music sequence *S* and hashes subsequences of length ℓ into a hash table, whose bins are integer counters (initially set to 0). During the traversal, when a path *P* of length ℓ is examined, M^{2}P checks the corresponding bin and if its counter is less than 2, it prunes the traversal for extensions of *P* as *P* cannot possibly be a repeating pattern. However, if the value of the counter is larger than (or equal to) 2, *P* may not necessarily be a repeating pattern, due to possible hash collisions in the corresponding bin. Therefore, hashing can only provide a filter to reduce the number of examined paths of length ℓ. It should be noticed that an analogous hashing technique has been used in the case of mining itemsets [33]. As the hashing technique performs satisfactory only for paths with small length, in our implementation we consider the value of ℓ to be equal to 3 and 4 (a separate hash table is maintained for each considered value of ℓ).

The second technique considers the impact of cycles within the graph *G*. Evidently, the elements of repeating patterns and MLRPs may not be distinct, thus vertices and/or edges of *G* may be visited more than once for the currently examined path (within the traversal procedure). Let us assume that a path *P* is a repeating pattern but its length is less than CML. Then, if *P* contains a cycle, by using the vertices and edges in the cycle for an appropriate number of times (i.e., to follow the cycle as many times as needed), *P* can be extended so as the length of this extension to become equal to CML. Moreover, due to Case 3 (described in Section 4.1), a large number of paths can be inserted in MLQ. For this reason, we enhance the basic form of M^{2}P previously described, in order to locate the existence of a cycle within the currently visited path and, when Case 3 holds for a path containing cycles, we first count its frequency before appending it to MLQ. Despite the fact that this technique may increase the number of intermediate paths the frequency of which is counted, it also prevents the excessive increase of the members of MLQ (frequency of which will have to be calculated at the end of the traversal procedure).

The two aforementioned optimizations have been found to improve substantially the performance of M^{2}P. For this reason, they have been incorporated to the basic form that was described earlier and are being used henceforth.

5 Performance evaluation

In support of the efficiency of the proposed algorithm, this section presents a number of experiments that have been performed. A concise description of the experimentation platform and data sets is also given followed by a performance analysis based on experimental comparison of the baseline approach, i.e., the modified HLC, and the proposed approach, M^{2}P.

5.1 Experimental set-up

All algorithms described have been implemented and performed on a personal computer with 933 MHz Intel Pentium III processor, 512 MB RAM, operating system MS Windows 2000, while the developing package utilized was MS Visual C++. The performance measure was the wall-clock time measured in milliseconds.

The data sets employed for the experiments include real as well as synthetic music objects. The real music objects originated from MIDI files acquired from the World Wide Web, converted from the MIDI format to melody strings by retaining only the pitch information. These music objects include classical works (The Four Seasons—Concerto 1 "Spring/La Primavera"—Allegro composed by A. Vivaldi and "Toreador" composed by G.Bizet) as well as modern pieces ("Tears in heaven" composed by E. Clapton), since different kinds of music contain different characteristics and lead to varying lengths of MLRPs. The object size of "Spring/La Primavera," "Toreador" and "Tears in heaven" is 8.292, 22.898, 5.786, respectively, and denotes the length of each note sequence. The note count of an object is the number of discrete notes the melody string contains and for the previously mentioned music objects the note count is respectively 50, 72 and 40. As far as the synthetic music object is concerned, following [20], they were generated with uniform note distribution, object size 1,000 notes, while the note count is variable.

5.2 Results

Initially, we considered real music objects and we focused on classic ones. Herein, we present results for the "Spring/La Primavera" music object with respect to its size (i.e., by varying the size of the object that we take into account each time [20]). The results on execution time are illustrated in Fig. 5a. Moreover, Fig. 5b depicts the length of the discovered MLRPs with respect to the object's size.

As expected, the execution time of both algorithms increases with increasing object sizes. This is due to two reasons. During the increase of the length of the MLRP (see Fig. 5b), both algorithms examine more levels, thus the cost increases. When the length of MLRP remains constant for increasing object size (e.g., for size larger than 800), although HLC and M^{2}P do not examine more levels, the processing within the levels becomes more costly (due to the increase in the number

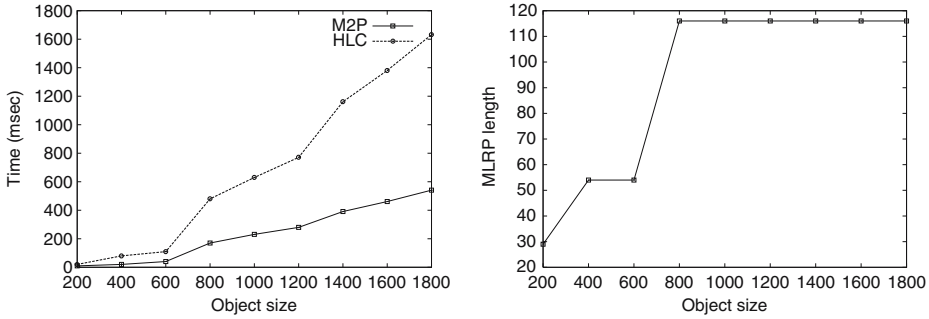


Fig. 5 Results for the classic music object: **a** execution time vs. object length; **b** length of MLRPs vs. object length

of intermediate repeating patterns). Nevertheless, M^{2}P clearly outperforms HLC by a factor of more than two in the case of larger object sizes.

In our next experiment we considered modern music objects. Herein, we present results from "Tears in heaven," which are depicted in Fig. 6. Particularly, Fig. 6a demonstrates the execution time for varying object size, whereas Fig. 6b the length of the discovered MLRPs again with respect to the object's size. Similarly to the case of classic music object, execution time for both algorithms increases with increasing object size. It worths noticing that the lengths of the discovered MLRPs (Fig. 6b) are relatively reduced compared to the case of the classic music object, supporting thus, the previously stated argument that different kinds of music contain different characteristics. Nevertheless, execution time shows no relative reduction (in the case of HLC it increases slightly), due to the increased number of intermediate repeating patterns (which is not shown). As in the previous experiment, M^{2}P compares favorably with HLC and presents an improvement for a factor up to 4 (for larger object sizes).

In order to examine the scalability of our method, we considered the "Toreador" music object, which is the largest among the music objects we considered. We varied the object size we considered each time and the largest size we examined was 20,000

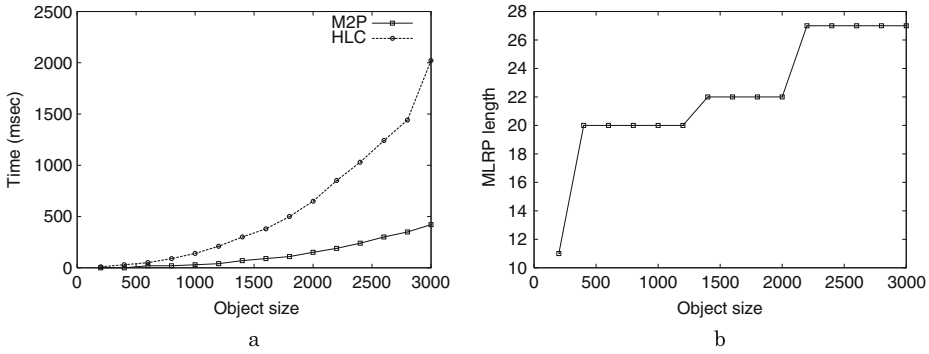


Fig. 6 Results for the modern music object: **a** execution time vs. object length; **b** length of MLRPs vs. object length

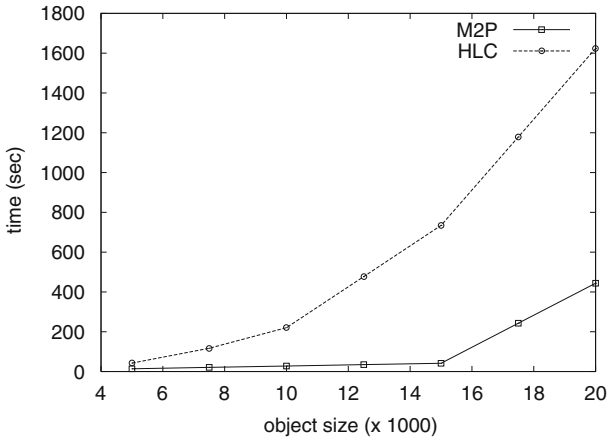


Fig. 7 Results for execution time (s) vs. object size ($\times 1,000$)

(close to the actual object's size). The results (in seconds) are illustrated in Fig. 7. As depicted, M^2P compares favorably to HLC.

We now move on to more clearly examine the impact of the length of discovered MLRPs on execution time. We used "Toreador" and varied its size so as to identify the points where an increase in the object's size leads to an increase in the length of discovered MLRP. Therefore, for the points found (expressed by the corresponding length of the discovered MLRPs) we measured the execution, and the results are depicted in Fig. 8a. As shown, the performance of M^2P is significantly better than the HLC, especially as the length of the MLRP increases. This fact illustrates that M^2P exhibits good scalability with respect to long patterns.

Next, we measured the impact of the note count. For this reason, we used synthetic music objects. The length of the objects was set to 1,000 notes and we varied the number of distinct notes (note count). The results with respect to the note count are presented in Fig. 8b. As expected, the execution time for both algorithm reduces for increasing note count. This is mainly due to the fact that the length of the repeating patterns and MLRPs tends to decrease as the note count increases for this type of

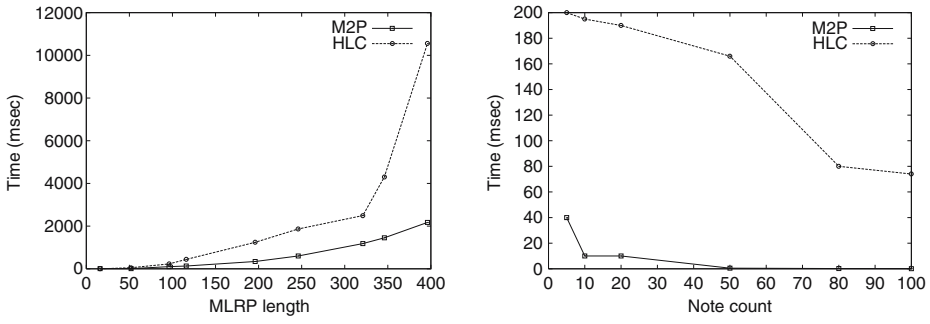


Fig. 8 Results for execution time vs.: **a** MLRP length (for a classic real music object); **b** note count (for synthetic music object)

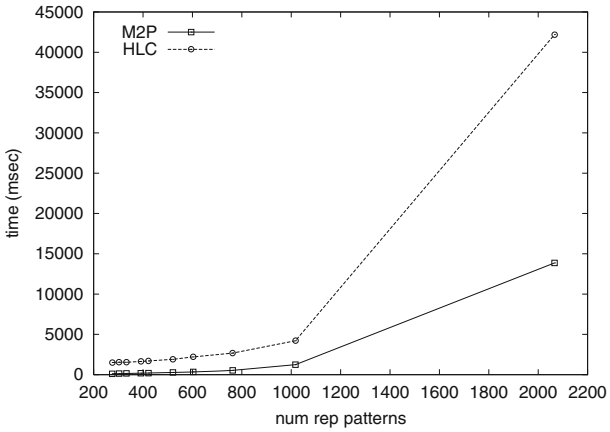


Fig. 9 Results for execution time vs. number of repeating patterns

music objects [20]. However, M^{2}P clearly outperforms HLC in all cases, verifying the results presented for real objects.

To further understand the behavior of M^{2}P, we examined its performance against the number of repeating patterns. As described, for a higher number of repeating patterns, HLC may require higher computation cost, since it has to examine more intermediate patterns before reaching the longest ones. Therefore, the number of repeating patterns in the music sequence can affect the performance difference between HLC and M^{2}P. Thus, for the sake of convenience and to easily control the number of repeating patterns within the same music sequence, we varied the frequency threshold. As expected, an increase in the frequency threshold results to less repeating patterns. In our measurement we used "Toreador" with length equal to 5,000 and varied the threshold in the range 1–10. The results are depicted in Fig. 9, where the *x*-axis illustrates the number of repeating patterns resulting from each frequency threshold (notice that the values are in inverse order, i.e., the lower numbers correspond to highest threshold values). As shown, as the number

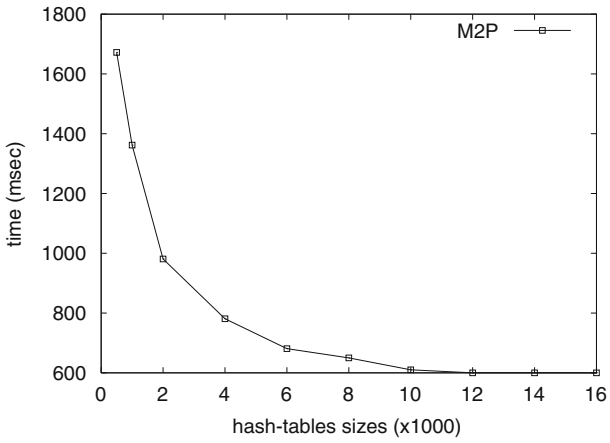


Fig. 10 Results for execution time of M^{2}P vs. hash-tables sizes

of repeating patterns increases, the performance difference between HLC and M^{2}P clearly increases. This is in accordance with the earlier described expectation.

Finally, we tested another factor that affects the performance of M^{2}P, the size of the hash tables that are used to optimize its performance. We used the "Spring/La Primavera" music object with length equal to 2,000 and varied the hash-tables sizes (i.e., the number of elements stored for each hash table). The results are given in Fig. 10. As expected, smaller hash-tables sizes result to less pruning and, thus, to higher execution times. On the other hand, there is a point up to which a further increase does not lead to adequate pruning, and the execution time does not decrease any further.

6 Conclusions

In this paper we introduced the problem of finding the maximum-length repeating patterns (MLRPs). This type of patterns helps in addressing the possible large number of plain repeating patterns in large music objects, and can be useful in discovering more sophisticated characteristics like music themes.

We present an efficient, novel algorithm, M^{2}P, for the extraction of MLRPs from music sequences comprising of pitch information. The efficiency of M^{2}P lays on the technique employed, which avoids costly repetition of frequency calculations by examining as few as possible intermediate repeating patterns, and aiming at fast reaching of MLRP set.

We have performed detailed experiments and measured several factors, such as the music object's size, the length of MLRP, the note count,the number of repeating patterns and hash-table size. The results indicate significant performance gains (up to a factor of four) compared to a prior method that was modified so as to constitute an efficient baseline algorithm.

As far as the possible future work is concerned, we will further consider the correspondence between repeating patterns, MLRPs, and themes. Moreover, as motives may contain a degree of variation within a single music piece, methods are required that will allow the discovery of music patterns in a more approximate way (i.e., not necessarily consecutive subsequences) [30].

References

1. Agrawal R, Srikant R (1995) Mining sequential patterns. In: Proceedings of the 11th IEEE international conference on data engineering (ICDE), Taipei, Taiwan, pp 3–14
2. Alghoniemy M, Tewfik AH (2000) User-defined music sequence retrieval. In: Proceedings of the 8th ACM international conference on multimedia, Los Angeles, CA, pp 356–358
3. Aucouturier J-J, Sandler M (2002) Finding repeating patterns in acoustic musical signals: applications for audio thumbnailing. In: Proceedings 22nd AES international conference on virtual, synthetic and entertainment audio, Espoo, Finland, pp 412–421
4. Bainbridge D, Bernbom G, Davidson MW, Dillon AP, Dovey M, Dunn JW, Fingerhut M, Fujinaga I, Isaacson EJ (2001) Digital music libraries—research and development. In: Proceedings of the 1st ACM/IEEE joint conference on digital libraries (JCDL), Roanoke, VA, pp 446–448
5. Barlow H, Morgenstern S (1975) A dictionary of musical themes. Crown, New York

6. Bartsch M, Birmingham WP, Bykowski D, Dannenberg RB, Mazzoni D, Meek C, Mellody M, Rand W, Wakefield GH, (2001) MUSART: music retrieval via aural queries. In: Proceedings of the 2nd annual international symposium on music information retrieval (ISMIR), Bloomington, IN, pp 73–81
7. Bayardo R (1998) Efficiently mining long patterns from databases. In: Proceedings of the ACM international conference on management of data (SIGMOD), Seattle, WA, pp 85–93
8. Byrd D, Crawford T (2002) Problems of music information retrieval in the real world. Inf Process Manag 38(2):249–272
9. Chavez E, Navarro G (2002) A metric index for approximate string matching. In: Proceedings of the 5th Latin American symposium on theoretical informatics (LATIN), New York, NY, pp 181–195
10. Chen ALP, Chang M, Chen J, Hsu J, Hsu C, Hua YS (2000) Query by music segments: an efficient approach for song retrieval. In: Proceedings of the IEEE international conference on multimedia and expo, New York, NY pp 873–876
11. Chen JCC, Chen ALP (1998) Query by rhythm: an approach for song retrieval in music databases. In: Proceedings of the workshop on research issues in data engineering (RIDE), Tucson, AZ, pp 139–146
12. Chen H, Chen ALP (2001) A music recommendation system based on music data grouping and user interests. In: Proceedings of the conference in information and knowledge management (CIKM), Hilton, Singapore, pp 231–238
13. Chuo T-C, Chen ALP, Liu C-C, (1996) Music DataBases: indexing techniques and implementation. In: Proceedings of the international workshop on multimedia databases management systems, Blue Mountain Lake, NY, pp 46–53
14. Crawford T, Iliopoulos CS, Raman R (1998) String matching techniques for music similarity and melodic recognition. Comput Musicol 11:73–100
15. Dovey MJ (2001) Adding content-based searching to a traditional music library catalogue server. In: Proceedings of the 1st ACM/IEEE joint conference on digital libraries (JCDL), Roanoke, VA, pp 249–250
16. Durey AS, Clements MA (2001) Melody spotting using hidden Markov models. In: Proceedings of the 2nd annual international symposium on music information retrieval (ISMIR), Berkeley, CA, pp 109–117
17. Francu C, Nevill-Manning CG (2000) Distance metrics and indexing strategies for a digital library of popular music. In: Proceedings of the IEEE international conference on multimedia and expo, New York, NY, pp 889–894
18. Hamilton JD (1994) Time series analysis. Princeton University Press, Princeton, NJ
19. Hsu J, Liu C, Chen ALP (1998) Efficient repeating pattern finding in music databases. In: Proceedings of the ACM international conference on information and knowledge management (CIKM), Bethesda, MD
20. Hsu JL, Liu CC, Chen ALP (2001) Discovering non-trivial repeating patterns in music data. IEEE Trans Multimedia 3(3):311–325
21. Iliopoulos CS, Kurokawa M (2002) Exact & approximate distributed matching for musical melodic recognition. In: Proceedings of the convention on artificial intelligence and the simulation of behaviour (AISB), Imperial College, London, UK, pp 49–56
22. Iliopoulos CS, Niyad M, Lenstrom K, Pinzon YJ (2002) Evolution of musical motifs in polyphonic passages. In: Proceedings of the convention on artificial intelligence and the simulation of behaviour (AISB), Imperial College, London, pp 67–75
23. Kang YK, Kim YS, Ku KI (2001) Extracting theme melodies by using a graphical clustering algorithm for content-based MIR. In: Proceedings of the 5th East-European conference on advances in databases and information systems (ADBIS), Springer-Verlag, London, pp 84–97
24. Kassler M (1966) Toward musical information retrieval. Perspect New Music 4(2):59–67
25. Koh JL, Yu WDC (2001) Efficient feature mining in music objects. In: Proceedings of the 12th conference in database and expert system applications (DEXA), London, UK, pp 221–231
26. Kornstadt A (1998) Themefinder: a web-based melodic search tool. Comput Musicol 11:231–236
27. Lin D-I, Kedem Z (2002) Pincer-search: an efficient algorithm for discovering the maximum frequent set. IEEE Trans Knowl Data Eng 14(3):553–566
28. Liu CC, Hsu JL, Chen ALP (1999) Efficient theme and non-trivial repeating pattern discovering in music databases. In: Proceedings of the 15th IEEE international conference on data engineering (ICDE), Sydney, Australia, pp 14–21
29. Meek C, Birmingham WP (2001) Thematic extractor. In: Proceedings of the 2nd annual international symposium on music information retrieval (ISMIR), Bloomington, IN, pp 119–128

30. Mongeau M, Sankoff D (1990) Comparison of musical sequences. Comput Humanit 24:161–175
31. Nishimura T, Hashiguchi H, Takita J, Zhang JX, Goto M, Oka R (2001) Music signal spotting retrieval by a humming query suing start frame feature dependent continuous dynamic programming. In: Proceedings of the 2nd annual international symposium on music information retrieval (ISMIR), Bloomington, IN, pp 211–218
32. O'Maidin DS, Cahill M (2001) Score processing for MIR. In: Proceedings of the 2nd annual international symposium on music information retrieval (ISMIR), Bloomington, IN, pp 59–64
33. Park J, Chen M-S, Yu P (1997) Using a hash-based method with transaction trimming for mining association rules. IEEE Trans Knowl Data Eng 9(5):813–825
34. Pienimaki A (2002) Indexing music databases using automatic extraction of frequent phrases. In: Proceedings of the 3nd annual international symposium on music information retrieval (ISMIR), Paris, France, pp 25–30
35. Pikrakis A, Theodoridis S, Kamarotos D (2002) Recognition of isolated musical patterns using hidden markov models. In: Proceedings of the II international conference on music and artificial intelligence (ICMAI), Edinburg, Scotland, pp 133–143
36. Raphael C (2001) Automated rhythm transcription. In: Proceedings of the 2nd annual international symposium on music information retrieval (ISMIR), Bloomington, IN, pp 99–107
37. Rolland P-Y, Ganascia J-G (2002) Pattern detection and discovery: the case of music data mining. In: Proceedings of the conference on pattern detection and discovery, London, UK, pp 190–198
38. Shifrin J, Pardo B, Meek C, Birmingham W (2002) HMM-based musical query retrieval. In: Proceedings of the 2nd ACM/IEEE-CS conference on digital libraries, Portland, OR, pp 295–300
39. Smith L, Medina R (2001) Discovering themes by exact pattern matching. In: Proceedings of the 2nd annual international symposium on music information retrieval (ISMIR), Bloomington, IN, pp 31–32
40. Takasu A, Yanase T, Kanazawa T, Adachi J (1999) Music structure analusis and its application to theme phrase extraction. In: Third European conference on research and advanced technology for digital libraries, Paris, France, pp 92–105
41. Uitdenbogerd A, Zobel J (1999) Melodic matching techniques for large music databases. In: Proceedings of the ACM international multimedia conference, Orlando, FL, pp 57–66
42. Velivelli A, Zhai C, Huang TS (2003) Audio segment retrieval using a synthesized HMM. In: Proceedings of the ACM SIGIR workshop on multimedia information retrieval, Toronto, Canada
43. Zaki M, Parthasarathy S, Ogihara M, Li W (1997) New algorithms for fast discovery of association rules. In: Proceedings of the international conference on knowledge discovery and data mining (KDD), Menlo Park, CA, pp 283–286

Ioannis Karydis was born in Athens, Greece in 1979. He received a BEng (2000) in Engineering Science & Technology from Brunel University, UK, an MSc (2001) in Advanced Methods in Computer Science from Queen Mary University, UK and a PhD (2006) in Mining and Retrieval Methods for Acoustic and Symbolic Music Data from the Aristotle University of Thessaloniki, Greece. Currently he is researching as a post-doc in Music Databases. His research interests include Music databases, music information retrieval (indexing & searching) and music object representation.

Alexandros Nanopoulos was born in Craiova, Romania, in 1974. Graduated from the Department of Informatics, Aristotle University of Thessaloniki, Greece, on November 1996, and obtained a PhD from the same institute, on February 2003. The subject of his dissertation was: "Techniques for Non Relational Data Mining." He is co-author of more than 25 articles in international journals and conferences, also co-author of the monograph "Advanced Signature Techniques for Multimedia and Web Applications." His research interests include spatial and web mining, integration of data mining with DBMSs, and spatial database indexing.

Yannis Manolopoulos was born in Thessaloniki, Greece in 1957. He received a B. Eng (1981) in Electrical Eng. and a PhD (1986) in Computer Eng., both from the Aristotle Univ. of Thessaloniki. Currently, he is Professor at the Department of Informatics of the latter university. He has been with the Univ. of Toronto, the Univ. of Maryland at College Park and the University of Cyprus. He has published about 200 papers in refereed journals and conference proceedings and has been honoured with over 1200 citations. He is co-author of a book on "Advanced Database Indexing," "Advanced Signature Indexing for Multimedia and Web Applications" both by Kluwer, "Nearest Neighbor Search: a Database Perspective", "R-trees: Theory and Applications" both by Springer Verlag. He served/serves as PC (Co-) Chair or General (Co-) Chair of the 8th National Computer Conference (2001), the 6th ADBIS Conference (2002), the 8th SSTD Symposium (2003), the 1st Balkan Conference in Informatics (2003), the 16th IEEE SSDBM Conference (2004), the 8th ICEIS Conference (2006), and the 10th ADBIS Conference (2006). His research interests include databases, data mining, web information systems and bibliometrics. Further information can be found at http://delab.csd.auth.gr/~manolopo .