An Efficient Algorithm for Mining Maximum-Length Repeating Patterns in Music Databases

Ioannis Karydis, Alexandros Nanopoulos, and Yannis Manolopoulos[⋆]

Dept. Informatics, Aristotle University,
Thessaloniki, 54124, Greece
emails: {karydis,alex,manolopo}@delab.csd.auth.gr

Abstract. Maximum-length repeating patterns (MLRP) are parts of the melody string of a music object, that appear frequently and have the largest length among all repeating patterns. In this paper we examine the problem of discovering MLRP in music objects. We present an algorithm for the extraction of MLRP, which discovers all maximum-length repeating patterns using an efficient accession during searching, by avoiding costly repetition frequency calculation and by examining as few as possible repeating patterns in order to reach MLRP. Experimental results illustrate the significant performance gains due to the proposed algorithm, compared to an existing baseline algorithm.

Keywords: content-based music retrieval, 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. Music data, due to their complex structure and their subjectivity to inaccuracies caused by perceptual and cognitive effects, introduce new challenges.

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 (c.f., Section 3). Their use (through the notion of motifs) has been extensive through out the history of music [3] as well as in the modern music [2], since they comprise a compact form for indexing the original formats, a reference point for the discovery of music or even *characteristic signatures* of music objects [8].

For the problem of efficient discovery of repeating patterns, recent research has employed data mining techniques [12, 16, 17, 24]. Focus has been oriented on the longest *non-trivial repeating patterns* [17, 12] which are typically those that can yield to themes [17] or themes are based upon [12](constrained by a maximum length value, e.g., 30).

A straightforward approach for the discovery of the longest repeating patterns is rather inefficient, since 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 for the efficient mining of

[⋆] Contact author: Tel: +302310996363, Fax: +302310998419 the longest repeating patterns, which will not have to undergo the discovery of many intermediate repeating patterns.

In this paper, we propose a novel algorithm that discovers all maximum-length repeating patterns (MLRPs) using a fast ascending, as far as the length of the patterns is concerned, during searching so as to quickly reach these patterns. Based on prior work on repeating patterns, we focus on note-sequences.

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).
- 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.

The rest of the paper is organised as follows. Section 2 describes the related work, whereas Section 3 discusses further one of the existing works that is used as a baseline algorithm. Section 4 describes the proposed method and Section 5 contains the experimentation results. Finally, Section 6 draws the conclusion.

2 Related Work

Early works on music information retrieval date back on 1966 [15]. Currently, many disciplines of music IR have been extensively 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) [1], pieces of structured or unstructured music [4, 10, 14, 25], humming [11, 9, 21, 4, 7] or even classic western musical notation [18] as queries.

Music is available in three basic representations: Audio, Time-stamped Events and Common Music Notation [6]. To achieve semantic, efficiency as well as to overcome data processing constraints the previously mentioned basic representations require a mapping. Although numerous approaches exist in the literature, the similarity of two mapped music objects mainly depends on the string matching core technique since, most usually, the mapping procedure for a music object produces a string of a chosen characteristic.

The process of mining repeating patterns is described in [12, 17], where two algorithms are proposed for the discovery of non-trivial repeating patterns and feature melody string. Koh and Yu ([16]) presented a means of mining the maximum repeating patterns from the melody of a music object using a bit index sequence. Rolland and Ganascia [23] described an algorithm for the mining of sequential patterns in music data, which considers several peculiarities of music objects.

As far as the use of repeating patterns in theme discovery is concerned, Smith and Medina [24] proposed a pattern matching technique that is based on a collection of previously found longest repeating patterns. Meek and Birmingham [19] developed several features that they consider for the discovery of themes. An interesting web-based system for theme discovery is presented in [13].

In the field of itemsets mining, several methods have been proposed recently for the discovery of the maximum-length frequent itemsets [5, 26]. There is distinct analogy between the problem examined in [5, 26] and the problem of discovering MLRPs. However, mining MLRPs presents important differentiations due to which the aforementioned approaches cannot be directly applied.

3 Background and Motivation

3.1 Definitions

Following previous research on discovering repeating patterns [12, 16, 24], we consider a music sequence to be a sequence of symbols from an alphabet containing discrete elements. The discrete elements correspond to pitches of music notes. The representation of the music sequence as a sequence of symbols corresponding to pitches is utilised for the easier presentation of the approaches introduced, as it is suggested in previous work [12] (notice that for the MIDI representation, the size of the alphabet is equal to 128).

Definition 1 (Repeating pattern) *Given a music sequence S, a repeating pattern P is a subsequence of consecutive elements of S that appears at least twice in S [12].*

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 [16]) *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: (i) X is a subsequence of X', and (ii) the $freq(X) = freq(X')$.*

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

For convenience, the maximum-length repeating patterns are henceforth denoted as MLRPs.

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 Motivation

Hsu et al. [12] 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) and will be used as a baseline algorithm for the extraction of MLRPs. HLC develops in two stages. In the first stage[1], repeating patterns of length 2^k (initially, $k = 0$) are found. The search proceeds 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 $2^{k_{l-1}} \leq L < 2^{k_l}$. More details can be found in [12].

As far as the HLC is concerned, among the other non-trivial repeating patterns, it discovers the set of all MLRPs, in a quite efficient way, due to the following reasons: (i) Only a logarithmic number of intermediate lengths is considered to discover the MLRPs, whereas a straightforward approach would check all possible lengths between 1 and L. (ii) Through our experimental measurements we have found that the most time consuming stage of the HLC is the second stage. 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 abovementioned 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, thus performing extra time-consuming frequency calculations. 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 has to be mentioned that [16] proposes a different approach for the discovery of repeating patterns. However this approach is not efficient for the purpose of discovering MLRPs.

4 The Proposed Method

4.1 Outline of the Approach

This section describes the proposed algorithm, 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 and RP[x] be the set of repeating patterns the length of which is x. Assume that we have identified all repeating patterns of length two, which are denoted as $\text{RP}[2] = \{\langle s_i, s_j \rangle : s_i, s_j \in S, \textit{freq}(\langle s_i, s_j \rangle) \geq 2\}$. 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 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]).

[1] The second stage is irrelevant to the discovery of MLRPs and is not commented.

Each path P in G is 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, leading 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 $\mathrm{M}^2\mathrm{P}$ is to identify in the aforesaid search space those paths that have maximum length and correspond to a repeating pattern. To attain this, $\mathrm{M}^2\mathrm{P}$ traverses G by searching for the paths that originate from any of its vertices. While encountering paths, $\mathrm{M}^2\mathrm{P}$ is concerned in identifying only these that are candidates to become a MLRP. During the traversal, it keeps track of the path C that has already been visited and: (i) has, so far, the maximum length, and (ii) corresponds to a repeating pattern (i.e., its frequency has been counted and found to be larger than two)[2]. 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 was considered during the traversal, so its frequency has been counted due to case 1).

Following the previously discussed approach, $\mathrm{M}^2\mathrm{P}$ 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 $\mathrm{M}^2\mathrm{P}$, 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 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 $\mathrm{M}^2\mathrm{P}$ is done by using a string matching

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

algorithm[3], 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 Figure 1. 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).

```
Procedure M$^2$P(MusicSequence $S$)
begin
    RP[2] = Find all rp with length 2
    Construct G(RP[2])
    CML := 2
    MLQ := $\emptyset$
    for each $v \in V(G)$
        Traverse($G$, $v$, $\langle v \rangle$, CML, MLQ)
    endfor
    for each $P \in$ MLQ
        if (CountFreq($q$)$\geq 2$)
            Output($P$)
        endif
    endfor
end
```

```
Procedure Traverse(Graph $G$, Vertex $v$,
    Path $P$, int CML, Queue MLQ)
begin
    bool prune := false
    Append($P$, $v$)
    if Length($P$) $>$ CML
        if CountFreq($P \geq 2$)
            MLQ := $P$
            CML = Length($P$)
        else
            prune := true
        endif
    else if length($P$) = CML)
        Enqueue(MLQ, $P$)
    endif
    if not prune
        for each $u \in V(G)$ and $\langle v \rightarrow u \rangle \in E(G)$
            Traverse($G$, $u$, $P$, CML, MLQ)
        endfor
    endif
end
```

Fig. 1. The MLRP algorithm.

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

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

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 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. For this example, $S = EBCDEHGABFJDEHGJEBCDEABFJ$, its RP[2] set and the corresponding graph G are illustrated in Figure 2. Assume (w.l.o.g.) that the M^{2}P begins its traversal from the paths emanating from vertex A and from edge AB in particular. Initially, path *ABC* is visited (Figure 3a). Since its length is $3 > CML = 2$, its frequency is counted for and found equal to 0. Therefore, M^{2}P 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 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.

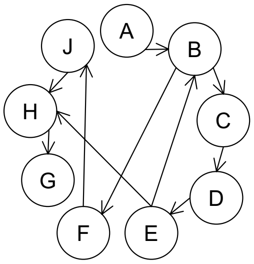
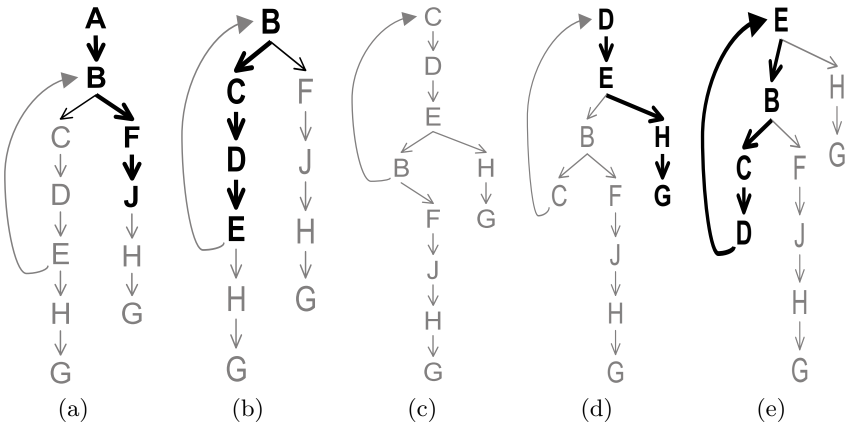


Fig. 2. The example graph G.



(a) (b) (c) (d) (e)

Fig. 3. Example of paths originating from vertices A, B, C, D, and E.

Next, the traversal moves on to vertex B (Figure 3b) 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 (Figure 3c) offer no change CML or MLQ, while the paths resulting from vertex D (Figure 3d) 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 (Figure 3e), the path $EBCD$ is added to the MLQ (MLQ={ABFJ,BDCE,DEHG,EBCD}). Next, path $EBCDE$ is examined, and its frequency is counted 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. 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 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, two techniques that we used to enhance the basic form of M^{2}P will now be described.

As indicated in [12], 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 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. 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 provide only 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 [22]. As the hashing technique pays-off 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). 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.

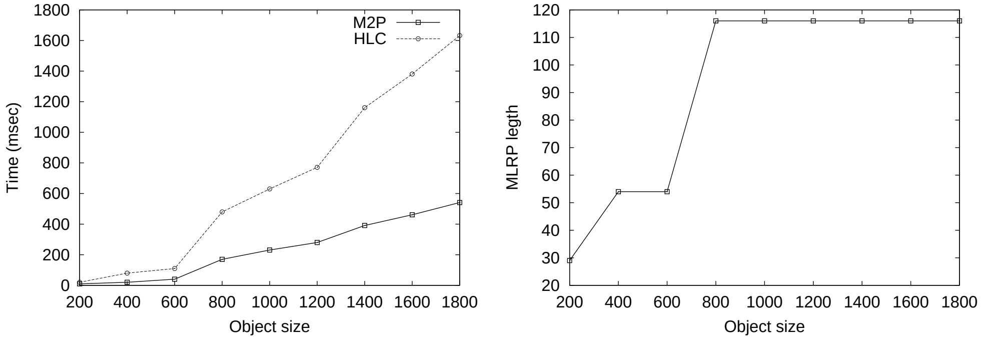
5.1 Experimental Set-up

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

The data sets employed for the experiments included real as well as synthetic music objects. The real music objects originate from MIDI files acquired from the World Wide Web, converted from the MIDI format to melody strings. These music objects include classical works (The 4 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 [12], the synthetic data is generated with uniform note distribution, object size 1000 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 [12]). The results on execution time are illustrated in Figure 4a. Moreover, Figure 4b depicts the length of the discovered MLRPs with respect to the object's size.



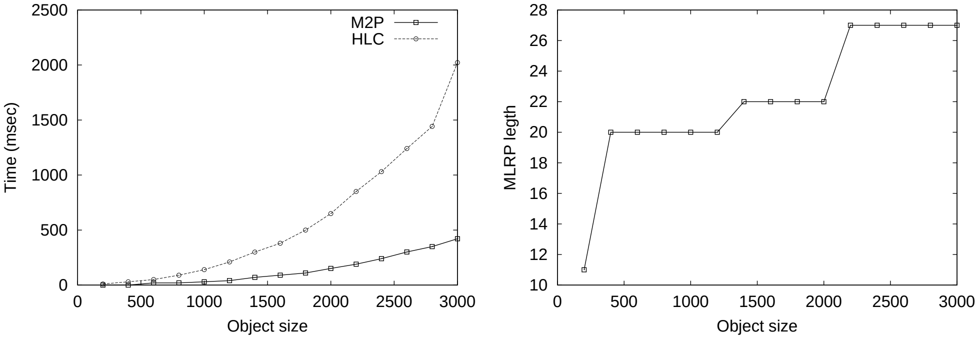
(a)

(b)

Fig. 4. Results for the classic music object: (a) Execution time vs. object length; (b) Length of MLRPs vs. object length.

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 Figure 4b), 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 of intermediate repeating patterns). Nevertheless, M^{2}P clearly outperforms HLC by a factor 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 Figure 5. Particularly, Figure 5a demonstrates the execution time for varying object size, whereas Figure 5b 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 (Figure 5b) 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).

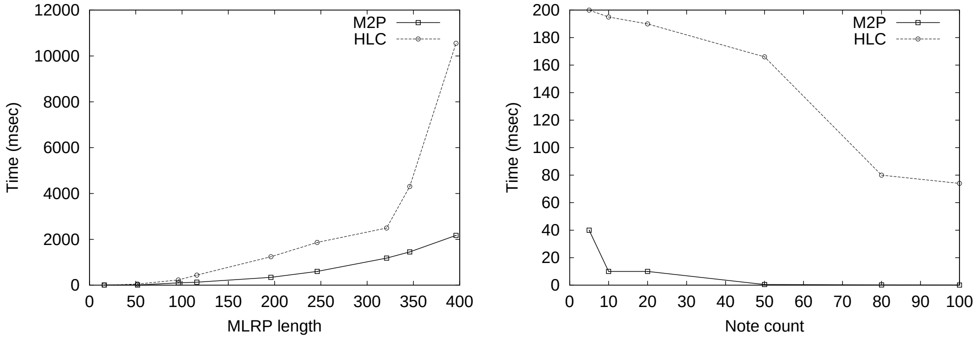


(a) (b)

Fig. 5. Results for the modern music object: (a) Execution time vs. object length; (b) Length of MLRPs vs. object length.

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 Figure 6a. As shown, the performance of M^{2}P is significantly better than the HLC especially as the length of the MLRP increases. This fact illustrates that M^{2}P presents good scalability with respect to long patterns.

Finally, 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 Figure 6b. 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 music objects [12].



(a) (b)

Fig. 6. Results for Execution time vs.: (a) MLRP length (for a classic real music object); (b) Note count (for synthetic music object).

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

6 Conclusions

In this paper we introduced the problem of mining 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 a 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 MLRPs.

We have performed detailed experimental results and measured several factors, like the music object's size, the length of MLRP, and the note count. The experimental 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. Regarding future work, we will further consider the correspondence between repeating patterns, MLRPs, and themes.

References

1. M. Alghoniemy and A.H. Tewfik: "User-Defined Music Sequence Retrieval", *Proceedings 8th ACM International Conference on Multimedia*, pp.356-358, 2000.
2. J.-J. Aucouturier and M. Sandler: "Finding Repeating Patterns in Acoustic Musical Signals: Applications For Audio Thumbnailing", *Proceedings 22nd AES International Conference on Virtual, Synthetic and Entertainment Audio*, 2002.
3. H. Barlow and S. Morgenstern: "A Dictionary of Musical Themes", *Crown*, New York, 1975.

4. M. Bartsch, W.P. Birmingham, D. Bykowski, R.B. Dannenberg, D. Mazzoni, C. Meek, M. Mellody, W. Rand and G.H. Wakefield: "MUSART: Music Retrieval Via Aural Queries", *Proceedings 2nd Annual International Symposium on Music Information Retrieval (ISMIR)*, 2001.
5. R. Bayardo: "Efficiently Mining Long Patterns from Databases", *Proceedings ACM International Conference on Management of Data (SIGMOD)*, pp.85-93, 1998.
6. D. Byrd and T. Crawford: "Problems of Music Information Retrieval in the Real World", *Information Processing and Management*, Vol.38, No.2, pp.249-272, 2002.
7. A.L.P. Chen, M. Chang, J. Chen, J. Hsu, C. Hsu and Y.S. Hua: "Query by Music Segments: an Efficient Approach for Song Retrieval", *Proceedings IEEE International Conference on Multimedia and Expo*, pp.873-876, 2000.
8. T. Crawford, C.S. Iliopoulos and R. Raman: "String Matching Techniques for Music Similarity and Melodic Recognition", *Computing in Musicology*, Vol. 11, pp.73-100, 1998.
9. M.J. Dovey: "Adding Content-Based Searching to a Traditional Music Library Catalogue Server", *Proceedings 1st ACM/IEEE Joint Conference on Digital libraries (JCDL)*, pp.249-250, 2001.
10. A.S. Durey and M.A. Clements: "Melody spotting using hidden Markov models", *Proceedings 2nd Annual International Symposium on Music Information Retrieval (ISMIR)*, pp.109-117, 2001.
11. C. Francu and C.G. Nevill-Manning: "Distance Metrics and Indexing Strategies for a Digital Library of Popular Music", *Proceedings IEEE International Conference on Multimedia and Expo*, pp.889-894, 2000.
12. J.L. Hsu, C.C. Liu and A.L.P. Chen: "Discovering Non-Trivial Repeating Patterns in Music Data", *IEEE Transactions on Multimedia*, Vol.3, No.3, pp.311-325, 2001.
13. D. Huron: "*Themefinder*". Available at www.themefinder.org.
14. Y.K. Kang, Y.S. Kim and K.I. Ku: "Extracting Theme Melodies by Using a Graphical Clustering Algorithm for Content-Based MIR", *Proceedings 5th East-Europena Conference on Advances in Databases and Information Systems (ADBIS)*, pp.84-97, 2001.
15. M. Kassler: "Toward Musical Information Retrieval", *Perspectives of New Music*, Vol.4, No.2, pp.59-67, 1966.
16. J.L. Koh and W.D.C. Yu: "Efficient Feature Mining in Music Objects", *Proceedings 12th Conference in Database and Expert System Applications (DEXA)*, pp.221-231, 2001.
17. C.C. Liu, J.L. Hsu and A.L.P. Chen: "Efficient Theme and Non-trivial Repeating Pattern Discovering in Music Databases", *Proceedings 15th IEEE International Conference on Data Engineering (ICDE)*, pp.14-21, 1999.
18. D.S. O'Maidin and M. Cahill: "Score Processing for MIR", *Proceedings 2nd Annual International Symposium on Music Information Retrieval (ISMIR)*, pp.59-64, 2001.
19. C. Meek and W.P. Birmingham: "Thematic Extractor", *Proceedings 2nd Annual International Symposium on Music Information Retrieval (ISMIR)*, pp.119-128, 2001.
20. M. Mongeau and D. Sankoff: "Comparison of Musical Sequences", *Computer and the Humanities*, Vol.24, pp.161-175, 1990.
21. T. Nishimura, H. Hashiguchi, J. Takita, J.X. Zhang, M. Goto and R. Oka: "Music Signal Spotting Retrieval by a Humming Query Suing Start Frame Feature Dependent Continuous Dynamic Programming", *Proceedings 2nd Annual International Symposium on Music Information Retrieval (ISMIR)*, pp.211-218, 2001.
22. J. Park, M.-S. Chen and P. Yu: "Using a Hash-Based Method with Transaction Trimming for Mining Association Rules", *IEEE Transactions on Knowledge and Data Engineering*, Vol.9, No.5, pp.813-825, 1997.
23. P.-Y. Rolland and J.-G. Ganascia: "Pattern Detection and Discovery: The Case of Music Data Mining", *Proceedings Conference on Pattern Detection and Discovery*, pp.190-198, 2002.

24. L. Smith and R. Medina: “Discovering Themes by Exact Pattern Matching”, *Proceedings 2nd Annual International Symposium on Music Information Retrieval (ISMIR)*, pp.31-32, 2001.
25. A. Uitdenbogerd and J. Zobel: “Melodic Matching Techniques for Large Music Databases”, *Proceedings ACM International Multimedia Conference*, pp.57-66, 1999.
26. M. Zaki, S. Parthasarathy, M. Ogihara and W. Li: “New Algorithms for Fast Discovery of Association Rules”, *Proceedings International Conference on Knowledge Discovery and Data Mining (KDD)*, pp.283-286, 1997.