Comparison of Dictionary-Based Approaches to Automatic Repeating Melody Extraction

Hsuan-Huei Shih, Shrikanth S. Narayanan and C.-C. Jay Kuo

Integrated Media Systems Center and Department of Electrical Engineering
University of Southern California, Los Angeles, CA 90089-2564
E-mail: {hshih,shri,cckuo}@sipi.usc.edu

ABSTRACT

Automatic melody extraction techniques can be used to index and retrieve songs in music databases. Here, we consider a piece of music consisting of numerical music scores (e.g. the MIDI file format) as the input. Segmentation is done based on the tempo information, and a music score is decomposed into bars. Each bar is indexed, and a bar index table is built accordingly. Two approaches were proposed to find repeating patterns by the authors recently. In the first approach, an adaptive dictionary-based algorithm known as the Lempel Ziv 78 (LZ-78) was modified and applied to melody extraction, which is called the modified LZ78 algorithm or MLZ78. In the second approach, a sliding window is applied to generate the pattern dictionary. It is called the "Exhaustive Search with Progressive LEngth" algorithm or ESPLE. Dictionaries generated from both approaches need to be pruned to remove non-repeating patterns. Each iteration of either MLZ78 or ESPLE is followed by pruning of updated dictionaries generated from the previous cycle until the dictionaries converge. Experiments are performed on MIDI files to evaluate the performance of the proposed algorithms. In this research, we compare results obtained from these two systems in terms of complexity, performance accuracy and efficiency. Their relative merits and shortcomings are discussed in detail.

Keywords: Music database, audio database, ESPLE, music indexing, repeating patterns, Lempel Ziv 78, LZ-78, melody search, pattern search

1. INTRODUCTION

Techniques for image and video feature extraction, indexing and retrieval have received a lot of attention recently in image and video database applications. A relatively small amount of effort has been put into audio feature extraction and indexing. Audio database management finds applications in music archiving, special effect sound search for audio editing, etc. Audio is also an integral part of multimedia databases, and often contains useful information for effective multimedia search. Multimedia database management with multi-modal information, such as audio, video and text, is an emerging trend. A better understanding of audio features and their utilization is an essential step towards creating a complete multimedia database management system. In the context of audio databases, music is especially important since it has become a commercial product in our daily life.

Some work has been done in music content analysis and database organization. Chen, et al. [1] proposed a pat-tree approach to index melodies, where pat trees were built with chords. The pat tree is a Patricia-like tree, which is a string containing all possible substrings of a given string. Ghias, et al. [2] used coarse melody contours as a key to query a music database. McNab, et al. [3], [4] used interval contours for interactive music retrieval. Tong and Kuo [5] considered a hidden Markov model (HMM) method to model special effect sounds for content-based audio query. Furthermore, Chen, et al. proposed the string-join approach [6] and the correlative matrix approach [7] to find repeating patterns in music. In the former approach, they repeatedly joined shorter repeating patterns to form a longer one. In the latter approach, they lined up a melody string in the x- and y-axis directions to form a correlative matrix and used that information to find repeating patterns. Both approaches use notes as the basic units. However, the computational complexity grows rapidly as the number of notes increases. Moreover, the essential duration information of each note was discarded in these systems.

In contrast, our proposed systems use bars, instead of notes, as the basic unit. Using bars not only captures the tempo information of melodies but also reduces the size of the input sequence. In our previous work, we focus on extraction of repeating patterns in the main melody of a given music piece. Repeating patterns can be used in organizing and indexing music databases. They also serve as an important feature for content-based retrieval. Furthermore, they can be used as a tool for analyzing characteristics of compositions and their composers. It is believed that people are particularly sensitive and receptive to certain salient portions in a piece of music. Here, we assume that repeating melodies constitute such a salient part. It is common that a piece of music is composed by certain small pieces of melodies that are repeated throughout the whole piece and can be memorized by people more easily. If a piece of music is written in music score form, repeating melodies in a piece of music are repeating patterns of notes in its music score.

We proposed two approaches to find these repeating patterns. The first one is a dictionary-based approach that relies on the classic work of Lempel and Ziv [8], [9] and is called the modified LZ78 algorithm or MLZ78. The second one is a sliding window approach called the "Exhaustive Search with Progressive LEngth" algorithm or ESPLE. In this work, we would like to compare their performances in terms of complexity, performance accuracy and efficiency. Their relative merits and shortcomings are discussed in detail.

The rest of this paper is organized as follows. In Section 2, some basic concepts and terminology from music theory to be used in this paper are described. The two algorithms under comparison are described in detail in Section 3. Experimental results are given in Section 4, and concluding remarks are presented in Section 5.

2. BASIC MUSIC THEORY

This section provides a brief review of some basic terms from music theory to be used in this paper.

2.1. Staff

A staff has five lines and four spaces as shown in Figure 1. Lower lines or spaces of a staff have lower pitches than higher lines or spaces. The lowest line of a staff is center 'Mi'. If a pitch cannot be drawn within these five lines and four spaces, extra lines can be added. As illustrated in Figure 1, if a center 'Do' is out of a staff's range, an extra line is drawn to represent it.

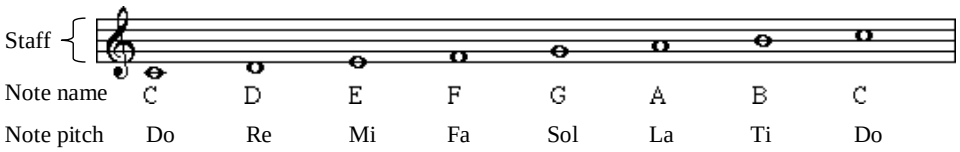

Figure 1: A staff and notes.

2.2. Note

A note is the basic unit in a music score. It contains pitch and duration information. A rest note is used to denote a rest period in a music score and contains only duration information. A note's pitch is shown by placing it on a staff. Figure 1 shows the pitch values and the corresponding names of the various notes. The duration of a note or a rest note is represented by a unique symbol. Figure 2. shows the symbols of notes and the relationships between different durations. For example, one whole note equals to two half notes.

2.3. Time Signature

A time signature is used to declare a time unit in a music piece and the number of time units comprising a music segment (referred to as a bar). Usually, a time unit is called a beat. A piece of music may contain more than one time signature. For example, a 3/4 time signature in Figure 3 implies that a quarter (1/4) note is a beat, and three beats form a segment (bar). The second segment of Figure 3. has 4 eighth notes and a quarter note, and the total is 3 beats.

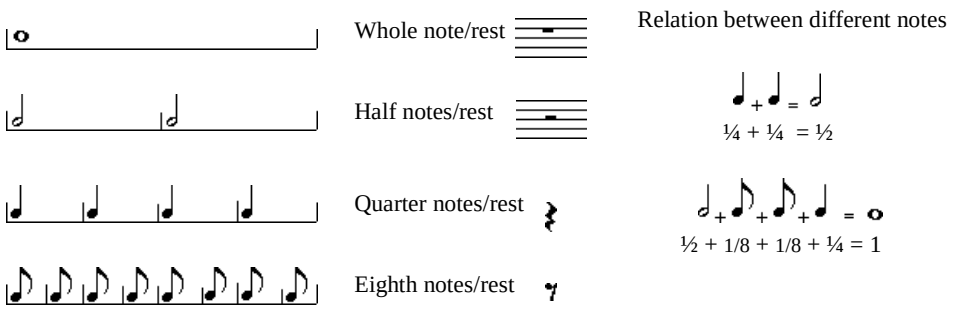

Figure 2: Notes' symbols and durations.

2.4. Bar

Music is often divided up into units called bars, as shown in Figure 3. The number of beats in a bar is based on a time signature. Bars under the same time signature have the same duration.

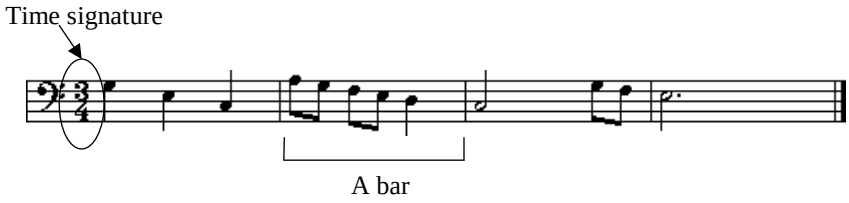

Figure 3: Time signature and bar.

3. TWO REPEATING PATTERN EXTRACTION SYSTEMS

3.1. Systems overview

Two dictionary-based repeating melody extraction systems are described in this section. Figure 4. is the functional flow-diagram of these two systems. The two main phases in the processing are "data preparation" and "repeating pattern extraction". The main difference of these two compared systems is the dictionary block. One used MLZ78[10] algorithm in the dictionary block and another used ESPLE[11] algorithm in the dictionary block. The detail differences of the dictionary block will be given later in this section. Music decomposition and bar indexing constitute the data preparation phase as described below.

1. The numerical music score is first segmented into bars based on the time signature.
2. A bar index table is built according to bars obtained from the segmented numerical music score.
3. Each bar is then replaced by its corresponding index, resulting in the so-called bar indexed music score, which is a term used through out this paper.
4. The bar indexed music score and the bar index table are ready for the next phase of processing.

The two main modules in the repeating pattern extraction phase are: dictionary processing and dictionary pruning. The extraction of repeating patterns is done iteratively. A repeating pattern list is introduced to store the extracted full-length repeating patterns. A repeating pattern is said to be of full-length if it is not a proper subset of any repeating pattern that has the same frequency count. A dictionary is generated after each dictionary iteration and pruned to remove non-repeating patterns. Moreover, extracted full-length

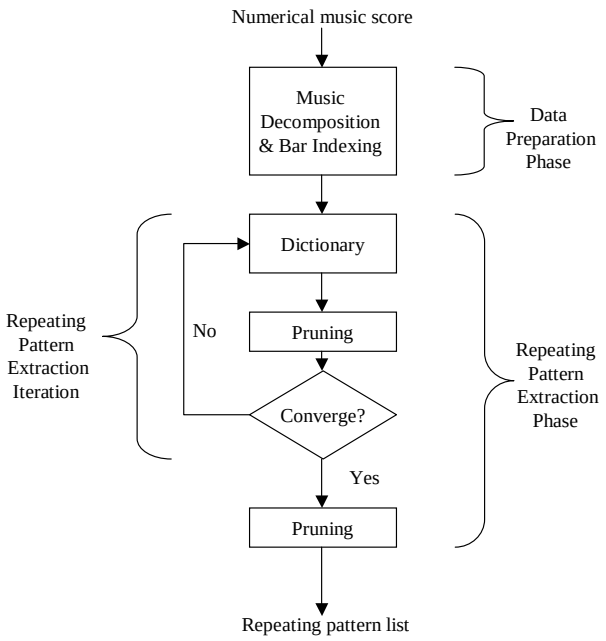

Figure 4: The system's functional flow-diagram.

repeating patterns are moved into a repeating pattern list. The pruned dictionary is passed on to the next dictionary iteration. The iteration is terminated when the system converges. Since these two systems use different dictionary algorithms, the convergent criteria are different. An ESPLE generated dictionary is said to be convergent when the pruned dictionary is empty. A MLZ-78 generated dictionary is said to be convergent when the pruned dictionary stop changing. Details of the repeating pattern extraction phase are given below.

1. The bar index table and the bar indexed music score generated from the data preparation phase are passed on to the repeating pattern extraction phase.
2. The initial dictionary and the repeating pattern list are empty before any repeating pattern extraction iteration.
3. A new dictionary is generated based on the bar-indexed music score and the pruned dictionary from the previous repeating pattern extraction iteration. (The first iteration utilizes only the initial dictionary).
4. The dictionary is then pruned to remove non-repeating patterns.
5. If a full-length repeating pattern is detected, it is moved from the dictionary to the repeating pattern list.
6. The pruned dictionary of the current iteration is checked for convergence. If it is indeed convergent, go to Step 8. If not, go to Step 7.
7. The pruned dictionary is passed to the next repeating pattern extraction iteration. Go to Step 3.
8. The repeating pattern list is pruned to remove parents' patterns. Then, the whole process is terminated and the repeating pattern list is returned.

Further details of each of the modules mentioned above are described in the following sections.

3.2. Music Decomposition and Bar Indexing

The bar (rather than the note) is used as a basic unit in our system due to the following considerations. First, a music note is too fine a unit to build a dictionary with since there are too many notes and their combinations in a piece of music, and the complexity of the dictionary building process grows very quickly. Second, as mentioned in Section 2.2, a note contains the pitch and the duration information. However, if a note is used as a symbol to build a dictionary, the duration of a note will be discarded. Therefore, bars are introduced to preserve the duration information of notes. In music scores, there are time signatures used to indicate the tempo of the underlying music, and a single piece of music may contain more than one time signature. In a music score, bars are used to group notes together according to a specified time signature. In our algorithm, bars are chosen to be the basic unit where a group of notes of the same time period are cascaded.

Usually, several bars form a repeating pattern. However, a repeating pattern may not start precisely at the beginning of a bar or stop at the end of a bar. In other words, they may start or stop at any note in a bar. For a given song, repeating patterns tend to start and stop at fixed positions in a bar. Let us consider an example in which a repeating pattern appears twice. There could be one or two bars that contain the repeating pattern's starting note. The time-offsets of these two starting notes in their respective bars are often the same while the offsets of music notes with respect to the whole piece of music are different. The same observation applies to the end point of a repeating pattern. The intermediate bars that lie between the starting and the ending bars are exactly the same. Just the leading and trailing bars of a repeating pattern require some special handling.

After decomposing a piece of music into bars, we should find a concise representation of each bar for further processing. To do so, a simple bar index table is generated. There are three attributes in this table: (1) a unique index number, (2) the bar pattern, and (3) the frequency of occurrence of this bar pattern. The index number is a sequence of non-negative integers, i.e. 0, 1, 2, 3, etc. The bar pattern is the sequence of pitch values of notes that are expressed numerically. The bar index frequency is the number of times a bar appears in a piece of music.

When the bar index table is built, the segmented music score is also concurrently converted into a bar indexed music score. This merely implies replacing each bar in the music score with its corresponding index. To facilitate the process of matching two bars, we can extract attributes from the bar pattern and perform attribute matching to filter out unlikely candidates. For example, the number of notes in a bar is an attribute that can be easily exploited. Furthermore, we can record pitch values of consecutive notes in a bar while ignoring their durations and, at the same time, discard all rest notes to derive another attribute. For this attribute, rests at the start, in the middle, or at the end of a bar are treated the same in the bar matching process. For example, let us consider a case where each note is a beat, and each bar has four beats. The two bars "Do-Re-Rest-Mi" and "Do-Re-Mi-Rest", where "Rest" means a rest note, will be treated as the same by the bar matching process. By making this assumption several different bars that have the same number of notes with the same pitch values will be matched to the same index in the bar index table. However, it should be noted that such combinations of notes occur rarely in the same piece of music. Even if it does happen, the system is designed to looking for repeating patterns. A non-trivial repeating pattern is a sequence of several bars. A single bar appearing in one music piece may quite likely appear in a different piece music. It usually does not contribute toward discrimination since it is quite difficult for people to identify a particular piece after listening to only one bar. Therefore, one bar is too short to be considered as a repeating pattern.

3.3. Dictionary Process

Followings are the algorithms of two different dictionary processes.

3.3.1. Modified Lempel Ziv 78[10]

The Lempel-Ziv 78 (LZ78) algorithm is a lossless compression scheme that has been widely used in text compression. A dictionary of variable length is constructed and adaptively updated by LZ78 while parsing a sequence of symbols. Vocabularies in the dictionary will be added according to the processed data. In our system, input symbols are bars with an appropriate index number and vocabularies in the dictionary are sequences of bar indices. Sequences of bar indices are called patterns through out this paper. The main idea of dictionary-based

compression is to detect longer vocabulary entries and encode them with shorter codewords. This process turns out to be a powerful tool in finding repeating patterns in music. The dictionary is the place where repeating patterns are accumulated.

"DA" is called the parent pattern of both "DAD" and "DAB". In general, to form a parent pattern that is N bars long by using LZ78 requires that the pattern appears at least N times in the underlying music. If N is large, it could be difficult to get a long parent pattern. A long parent pattern is needed for longer repeating patterns. To overcome this difficulty, we pass the same music piece through the LZ78 dictionary building system several times, and the dictionary in each LZ78 iteration is built based on the previously built dictionary.

The flow diagram of the modified LZ78 (MLZ78) is shown in Figure 5, and explained below.

1. A buffer and a dictionary are needed in MLZ78, and at the beginning they are both empty. The buffer is referred to as old/new word.

2. One new character is read in from the incoming data. If the bar index frequency is 1, empty the old/new word buffer and start from Step 2 again. Otherwise, go to the next step.

3. Append the new character to the old word, and it becomes the new word. There is at least one character in the new word buffer.

4. There are two cases. (a) If the new word is already in the dictionary, then this new word becomes the old word (and nothing is changed in the buffer). Start from Step 2 again. (b) If the new word is not in the dictionary, add the new word to the dictionary, empty the buffer and return an empty old word. Then, record the index of newly added pattern's parent. Start from Step 2 again.

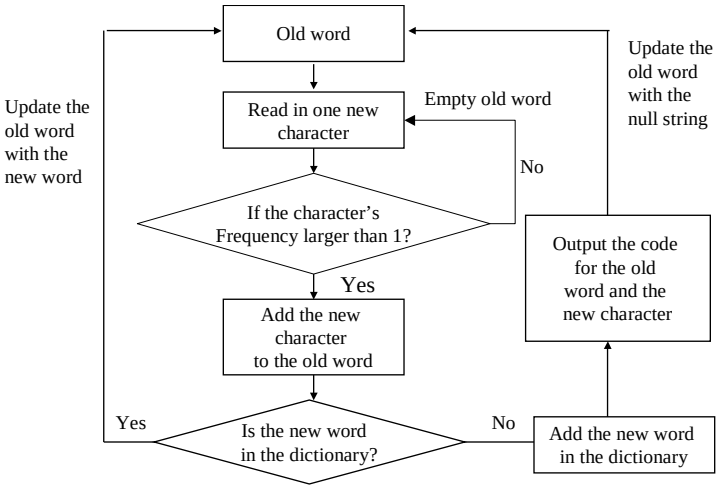

Figure 5: The block diagram of MLZ78.

Patterns in the dictionary may not be repeating patterns, and furthermore all parents of patterns are also included in the dictionary. The dictionary will diverge if these non-repeating patterns and pattern's parents are not handled properly. Hence, pruning the dictionary after each modified LZ78 iteration is essential to have a convergent dictionary, thus enabling easier extraction of repeating patterns. Details of the pruning techniques are discussed in section 3.4.

3.3.2. ESPLE[11]

The basic idea of ESPLE is as follows. A sliding window is applied to an input string, and the size of the window increases with increasing number of iterations. However, the window will not be applied to known non-repeating pattern portion of an input string. The windowed portion of the string will be checked for repeating patterns and the dictionary will be updated.

Before describing the ESPLE algorithm,the format of a dictionary should be explained. A dictionary contains 5 attributes: (1)the dictionary index number, (2)patterns, (3)the frequency, (4)the pattern length,and (5)positions of the pattern. The dictionary index number is a unique number for each pattern in the dictionary. A pattern of combination of several bars is recorded in the pattern attribute. The frequency is used to record the number of times a pattern appears. The pattern length is used to keep track of the length of pattern. Positions of a pattern are used to keep track of the pattern's appearance in an input string. First, instead of applying MLZ78 once in an iteration for an input string, a pattern extension is applied to multiple substrings which are generated from the input string. If an input string is ordered from the left to the right, multiple substrings are generated by shifting the starting point to the right for each position in the input string. For example, if the input string is "ABCDE", then the multiple input substrings are "ABCDE", "BCDE", "CDE", "DE",and "E". Each iteration only extends the length of the patterns by one. For the same example used previously, "A", "B", "C", "D", and "E" are the patterns after the first iteration. "AB", "BC", "CD", and "DE" are the newly generated patterns after the second iteration. Since the frequency of each index is recorded in the bar index table when it is first built (see Section 3.2) it is available prior to the execution of the ESPLE algorithm. This bar index frequency information can be used to improve the effectiveness of ESPLE. If the bar index frequency is one, clearly the corresponding index number should not be a part of any repeating pattern. Hence,such non-repeating bars should be ignored from being included in any of the extended patterns. Therefore,any pattern that is going to include a non-repeating bar should be discarded. In other words,in the pattern expanding process, a pattern in the old word buffer should stop expanding into longer patterns, when a non-repeating bar is read in. A "pattern frequency" attribute is included in the dictionary. It is used to count the number of times each pattern appears. The pattern frequency will be updated in the pruning phase. Non-repeating patterns will be removed from the dictionary.

The flow diagram of ESPLE is shown in Figure 6, and described below.

1. A windowing buffer is needed in ESPLE for each substring,and only one dictionary is required for all ESPLE iterations. Initially windowing buffers' size for different substrings are 1 character and the dictionary is empty.

2. One new character is read in from the incoming substring.If the bar index frequency is 1,terminate ESPLE iteration for the current substring.There is no more ESPLE iteration for this substring in the future iteration.

3. Increase the size of the window by 1 character, and the size of the window becomes to $n + 1$ where n is the size of the window of previous iteration. Apply the new expended window to the input sub string and generate a new word.

4. There are two cases.(a) If the new word is already in the dictionary,then update the corresponding word's frequency and position in the dictionary. (b) If the new word is not in the dictionary, add the new word to the dictionary. Then, record the pattern length and appearance position of the newly added pattern.

3.4. Pruning

Figure 7 gives the flow diagram of the pruning algorithm. The pruning phase has three stages: repeating pattern verification, repeating pattern extraction, and pattern elimination. In the first stage (i.e. repeating pattern verification), the repetition of each pattern is verified by using the frequency attribute in the dictionary. As mentioned in the previous section, non-repeating patterns will also be caught in the dictionary, so verifying repetitions becomes necessary. Although the proposed dictionary algorithms try to eliminate the problem of

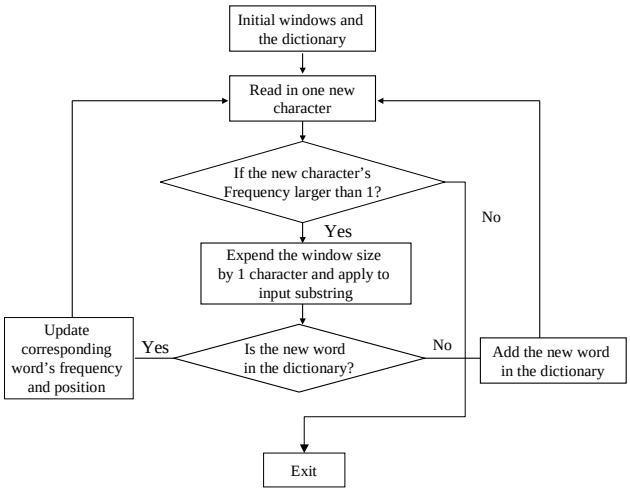

Figure 6: The block diagram of ESPLE.

having non-repeating patterns in the dictionary, some non-repeating patterns may still appear in the dictionary. The main reason for this phenomenon is a bar index that is not a part of any repeating patterns may have multiple appearances in a piece of music. These appearances typically occur in isolated places and the dictionary algorithms have no way of detecting this problem. Since the dictionary converges, the time required for checking repetitions will not grow as the number of iterations increases. All patterns in the dictionary will be checked for their repetition, while at the same time the frequency attribute in the dictionary will be updated.

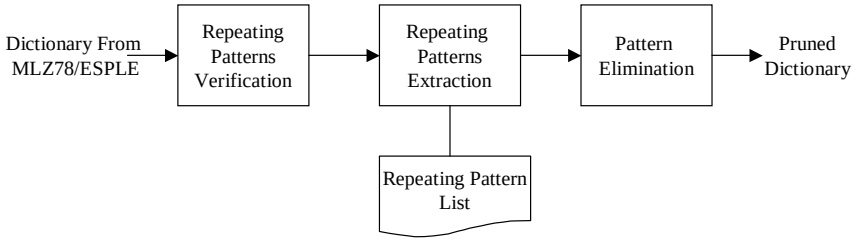

Figure 7: The block diagram of the pruning phase.

In the repeating pattern extraction stage, full-length repeating patterns are extracted and moved from the dictionary to a repeating pattern list. That is, entries of detected full-length repeating patterns are no longer retained in the dictionary. As mentioned previously, a repeating pattern is said to be a full-length repeating pattern, if it is not a proper subset of any repeating pattern which has the same frequency. A subset of a full-length repeating pattern may have a higher frequency than the full-length repeating pattern. Then, this subset may be another full-length repeating pattern if it is not a proper subset of other repeating patterns that have the same frequencies. All entries in the repeating pattern list are full-length repeating patterns. A threshold will be set to tell the system what the minimum length of a pattern should be for it to be considered as a repeating pattern. In practice, the threshold cannot be set too short or too long, since a too short pattern cannot be considered as a valid repeating pattern, and a too long pattern may cause no repeating patterns to be found. There is a way of determining repeating patterns by detecting non-repeating patterns. A pattern of length N is not a repeating pattern but its parent patterns of length N-1 must be repeating patterns. Otherwise, the non-repeating patterns will not be able to extend to length N.

In the pattern elimination stage, non-repeating patterns are removed from the dictionary. Patterns of length

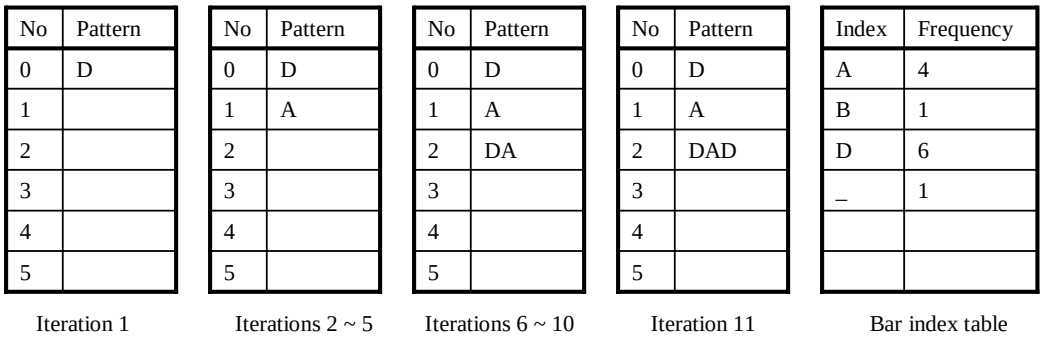

Figure 8: Bar index table and dictionaries for example of "DAD_DADABDAD" applying MLZ78.

equal to one are also removed. Extracted repeating patterns are also removed. Moreover, proper subsets of an extracted full-length repeating pattern will be eliminated as well. Since some patterns are removed from the dictionary, the indices of patterns will be reordered in the pruned dictionary. The consistency of parents' indexes will be updated in the pruned dictionary. Then, this pruned dictionary will be used in the next dictionary iteration, until the pruned dictionary convergence.

4. COMPARISON OF MLZ78 AND ESPLE

The two different approaches lead to different performance characteristics and slightly different results. However, each approach has its own advantages, and disadvantages.

4.1. String Propagation

The first difference is how each repeating pattern in the input string is growing via the application of these algorithms. In MLZ78, the expansion is done sequentially until the end of each string, and the pattern is expanded by only one element at one time whenever it is encountered in the string. When we reach the end of the string, we usually have to scan from the beginning again and repeat the same process several times until the dictionary converges. Unlike MLZ78, ESPLE perform pattern growing through the entire string in each iteration. However, this does not imply that MLZ78 is more efficient than ESPLE. Our experiments have shown that MLZ78 may need to go through a string several times before its convergence, while ESPLE will not grow non-repeating portions of the string at all. ESPLE only grows each substring once, and terminates its further expansion when a non-repeating bar index arrives or the end of substrings is reached. The number of substrings which ESPLE needs to grow decreases as the window size increases.

Let us consider an example of input sequence "DAD_DADABDAD". The dictionaries of different iteration stages of MLZ78 and ESPLE are shown in Figures 8 and 9, respectively. The number of iterations for MLZ78 to have the repeating pattern "DAD" in the dictionary is 11, and the number of iterations for ESPLE to have the repeating pattern "DAD" in the dictionary is 7. For detailed steps of getting example dictionaries, please refer to [10] and [11]. In fact, which approach is faster highly depends on the input string. If the length of a repeating pattern is short, or if the number of the repeating pattern appearance is larger than the length of the repeating pattern, then MLZ78 should have a better performance in terms of speed. On the other hand, if a long repeating pattern appears relatively few times in the string, ESPLE will have a better performance in speed.

4.2. Accuracy

In terms of accuracy, these two approaches produce slightly different results since they adopt different ways to find repeating patterns. For example, it is very difficult for human to decide the repeating pattern in an input string such as "ABCDABCDABCDAB". The repeating pattern could be "ABCD", "ABCDAB", or "CDAB". Therefore, different approaches may offer different results, and the result is highly dependent on the position

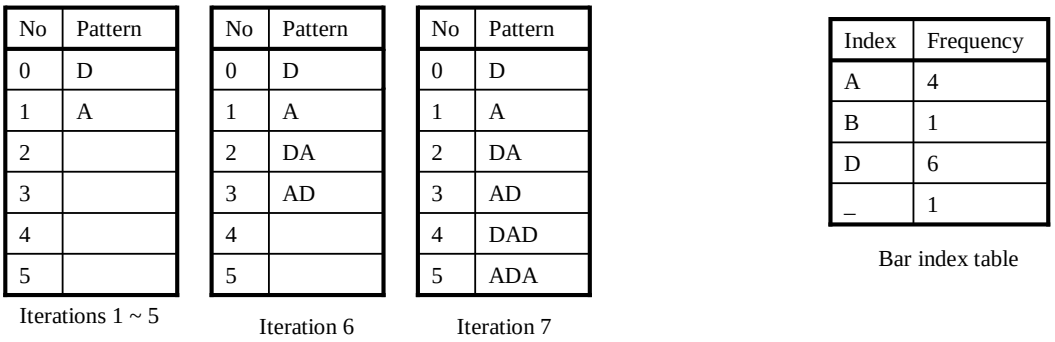

Figure 9: Bar index table and dictionaries for example of "DAD_DADABDAD" applying ESPLE.

where a repeating pattern appears, especially relative to other repeating patterns' positions. Table 1 shows slightly different results of MLZ78 and ESPLE.

Table 1: The results of different note insertion log probability

Title	MLZ78		ESPLE	
	NO. of iteration	NO. of repeating pattern	NO. of iteration	NO. of repeating pattern
Yellow Submarine	15	5	18	5
All I have to do is dream	4	3	5	3
500 miles	6	4	6	5
Hotel California	21	2	15	2

4.3. Subjective testing

An important issue is the relation between subjective user judgments obtained through listening experiments and extracted repeating patterns. It turns out that some of algorithmically extracted patterns are not sound like a complete repeating pattern to a human, or are longer than expected. Because the starting point and the ending point of a repeating pattern may not start from the first note of a bar and stop at the last note of a bar.

Hence, it is crucial to incorporate constraints from human hearing and music theory, to further refine the usefulness of the extracted repeating patterns. The application of such rules may improve subjective results for both systems. For example, after applying music rules specifying a melody's starting and ending, bar indices which meet these rules are marked as the potential starting bar and the potential ending bar. When extracting repeating patterns, the algorithms can only find repeating patters that satisfy these starting/ending bar indices. After repeating patterns are found, notes which are not part of repeating patterns are removed base on music theory.

To find the rules of the starting and the ending points requires thorough study of various kind of music. To simplify complicated music theory, a supervised training was introduced. Users subjectively label starting and ending points of repeating patterns extracted form MLZ78 and ESPLE, and information of starting and ending points were store in the starting/ending point lookup table. 150 songs were used to build the lookup table. 50 songs which were different from training songs were chosen for experiments. Let us refer experiments using the lookup table as new experiments. All experiments were done for both MLZ78 and ESPLE approaches.

When comparing the new results with to previous results, we see that the average iteration numbers of MLZ78 and ESPLE were slightly reduced after introducing the lookup table. That was so because dictionaries converged faster after setting a limit on some repeating patterns. The change in the number of extracted repeating patterns was also observed in some songs. Subjective user listening was applied in the experiments,

and users were asked to vote between new and previous experiments based on the their subjective judgments. The voting showed more satisfaction for the extracted repeating patterns by using the lookup table than those without the lookup table for most songs. However, we also notice that, since the starting/ending point lookup table was generated from 150 songs, this table is still not able to cover all the possible staring and ending points of another 50 songs.

5. CONCLUSION AND FUTURE WORK

In this research, we considered a piece of music consisting of numerical music scores (e.g. the MIDI file format) as the input. Segmentation was done based on the tempo information, and a music score was decomposed into bars. Each bar was indexed, and a bar index table was built accordingly. Two approaches, i.e. MLZ78 and ESPLE, to find repeating patterns were compared. Experiments were performed on MIDI files to evaluate the performance of the proposed algorithms. We compared results obtained from these two systems in terms of complexity, performance accuracy and efficiency. Their relative merits and shortcomings were discussed in detail.

In the future, we would like to work on supervised training to obtain the starting and the ending point lookup tables, and further improve the performance of unsupervised training. Several simple music rules that help to indicate the starting and/or the ending points of melodies will be embedded in unsupervised training. Subjective tests performed by music trained and non-music trained users will be compared against each other.

We would also like to continue our work on efficient extraction techniques to enhance obtained results. Since MIDI files are used as the input to our system, the bar representation used for pattern extraction is unambiguous. However, when a piece of music is either played or sung by people, we have to convert the acoustic waveform to the bar representation in a preprocessing step. This demands robust signal processing techniques. Besides, since the bar representation may not be as accurate as that obtained from MIDI files, we have to develop a matching process that permits a certain level of error tolerance. This may require statistical approaches to music pattern extraction.

REFERENCES

1. A. L. P. Chen and C. C. Liu, "Music databases: indexing techniques and implementation," in *Proceedings IEEE Intl. Workshop on Multimedia Data Base Management Systems*, 1999.
2. A. Ghias, J. Logan, D. Chamberlin, and B. C. Smith, "Query by humming: musical information retrieval in an audio database," in *Proceedings of ACM Multimedia Conference'95*, (San Francisco, California), November 1995.
3. R. J. McNab, L. A. Smith, I. H. Witten, C. L. Henderson, and S. J. Cunningham, "Towards the digital music library: Tune retrieval from acoustic input," in *In Digital Libraries Conference*, 1996.
4. R. J. McNab, "Interactive applications of music transcription," Master's thesis, Department of Computer Science, University of Waikato, New Zealand, 1996.
5. T. Zhang and C.-C. J. Kuo, *Content-based Audio Classification and Retrieval for Audiovisual Data Parsing*, Kluwer Academic Publishers, 2001.
6. C. C. Liu, J. L. Hsu, and A. L. P. Chen, "Efficient theme and non-trivial repeating pattern discovering in music databases," in *Proc. IEEE International Conference on Data Engineering*, 1999.
7. J. L. Hsu, C. C. Liu, and A. L. P. Chen, "Efficient repeating pattern finding in music databases," in *Proc. ACM Seventh International Conference on Information and Knowledge Management (CIKM)*, 1998.
8. J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," in *IEEE Transactions on Information Theory, number 3* **volume 23**, pp. 337–343, September 1977.
9. J. Ziv and A. Lempel, "Compression of individual sequences via variable-rate coding," in *IEEE Transactions on Information Theory, number 5* **volume 24**, pp. 530–536, September 1978.
10. H.-H. Shih, S. S. Narayanan, and C.-C. J. Kuo, "A dictionary approach to repetitive pattern finding in music," in *2001 IEEE International Conference on Multimedia and Expo (ICME2001)*, August 2001.

11. H.-H. Shih, S. S. Narayanan, and C.-C. J. Kuo, "Music indexing with extracted main melody by using modified lempel-ziv algorithm," in *International Symposium on the Convergence of IT and Communiactions (ITCOM 2001)*, August 2001.