멜로디 시퀀스를 이용하는 내용 기반 음악 검색 알고리즘

위조민[0] 구경이 김유성

인하대학교 전자계산공학과

zuminwei@yahoo.com g9721046@inhavision.inha.ac.kr yskim@inha.ac.kr

A Content-Based Music Retrieval Algorithm Using Melody Sequences

Zu-Min Wei[0] Kyong-I Ku Yoo-Sung Kim

Dept. of Computer Science & Engineering, Inha University, Incheon 402-751, Korea

ABSTRACT

With the growth in computer and network technologies, some content-based music retrieval systems have been developed. However, their retrieval efficiency does not satisfy user's requirement yet. Of course users hope to have a more efficient and higher precision for music retrieval. In this paper, so for these reasons, we propose an efficient content-based music retrieval algorithm using melodies represented as music sequences. From the experimental result, it is shown that the proposed algorithm has higher exact rate than the related algorithms.

1. INTRODUCTION

With the growth in digital representation of music, and of music stored in these representations, it is increasingly interested to search collections of music. So retrieving music from a collection of musical scores is essentially a matter of matching input strings against a database. This is a familiar problem in information retrieval, and so many efficient algorithms for finding substring in a body of text have been proposed.

Unfortunately, there are some problems with seeking an exact string-matching algorithm between the transcribed melodies and the music database. The first is the variability in the way that music is performed. For example, folk songs appear in many variants, while popular songs and well-known standards songs are often performed differently from the way they are notated. And then this is retrieval for music, not for text. So if an exact string-matching algorithm is used for music retrieval, it may be insufficient for application in tonal music as it disregards tonal qualities of pitches and pitch intervals. For example, a tonal transposition from a major key to a minor key result in a different encoding of the musical passage and thus exact string matching cannot detect the similarity between the two passages.

For it, we account it is necessary to allow approximate string matching [5] on the score database in order to retrieve music.

Also we know that approximate string matching algorithms have been proposed like δ-approximate string matching and γ-approximate string matching [2]. These kinds of string matching, however, are not efficient well yet. Especially, when δ value is larger, exact rate of retrieval will be low by the single δ-approximate string matching.

In this paper, so for reasons mentioned above, we will present a more efficient approximate string-matching algorithm named (δ, γ)-approximate string-matching algorithm for numeric string which is transcribed melodies. We will unite δ-approximate string matching and γ-approximate string matching well, and then obtain a more efficient and more perfect result for music retrieval, especially for larger δ values and larger melody corpuses.

In the rest of this paper is organized as follows. As the related work, drawbacks of traditional algorithm and musical content-based Information Retrieval System are introduced in section 2. The basic string definition, the (δ, γ)-approximate string matching algorithm and its processing are described in section 3 The experimental result is discussed in section 4. Finally, the conclusion will be stated in section 5.

2. RELATED WORK

2.1 Drawbacks of Traditional Algorithm

We know that some approximate string matching algorithms have been proposed like δ-approximate string matching and γ-approximate string matching. They, however, still have some shortcomings. If δ is larger, the efficiency of retrieval may be lower. Especially, in a larger music database, it may be found many irrelevant results. For example, we use δ-approximate string matching to find two matching (3,4,6,2) and (4,5,7,1) for the query (3,4,6,2), δ=1(see Table 1 and 2). From the two results, however, we can note that the second result is irrelevant with the query. So, the result is an inappropriate result for user's query. Of course, this has been proved by the experiment. Also retrieval by the γ-approximate string matching is not very efficient also.

To overcome these drawbacks and obtain a more efficient retrieval, we propose an efficient algorithm that joins δ-approximate string matching and γ-approximate string matching.

Following, we will describe the related environment for music retrieval.

2.2 Content-Based Music Retrieval System

In content-based music retrieval systems, user query is in humming melody instead of text.

The system consists of components as follows:

· A web browser-based user interface. User inputs his query through it.

· *Transcription. User's humming is transformed into a string* of the pitch interval.

· Searching Algorithm. Using proposed algorithm searching music objects matching the user's query.

· Music Database. It stores the music objects.

In generally, user's humming query is translated into a string of pitch values, and then by using the searching algorithm, the *appropriate music objects are retrieved as the result from music* database.

3. (δ, γ)-APPROXIMATE STRING MATCHING ALGORITHM

3.1 Basic String Definition

A string is a sequence of symbols from an alphabet Σ; A string x of length n is represented by $x_1..x_n$, where $x_i \in \Sigma$ for $1 \le i \le n$. Let Σ be an alphabet of integers and δ is an integer.

Two symbols a, b of Σ are said to be δ-approximate, denoted $a =_\delta b$, if and only if

$$|a - b| \le \delta \qquad (1\text{-}1)$$

We say that two string x, y are δ-approximate, denoted $x =_\delta y$, if and only if

$$|x| = |y| \text{ and } x_i =_\delta y_i,\ \forall_i \in \{1..|x|\} \qquad (1\text{-}2)$$

For a given integer γ we say that two strings x, y are γ-approximate, denoted $x =_\gamma y$, if and only if

$$|x| = |y| \text{ and } \sum_{i}^{n} |x_i - y_i| < \gamma \qquad (1\text{-}3)$$

Furthermore, if and only if x and y satisfy condition (1-2) and (1-3), we say that two strings x, y are (δ, γ)-approximate, denoted $x =_{\delta,\gamma} y$. This approximate string-matching algorithm is just presented for this paper.

3.2 (δ, γ)-Approximate String Matching Definition and Its Processing

The algorithm is based on O(1) time computation of the "Delta States" $DState_r$ and the "Gamma States" $GState_r$. It is by using bit operation under the assumption that m less or equal than the number of bits in a machine word.

Given a string $S = s_1..s_n$, A pattern $P = p_1..p_m$; then compute all position r of S such that $P =_{\delta,\gamma} S[r...r+m-1]$, where $r \in \{1 \cdots n\}$. The detailed steps of the algorithm as follows:

1. Firstly, we need compute "Delta Table" DT. Set DT[e]=f, where e denotes a symbol occurring in S and $f=f_1..f_m$ is a binary word, if $|e - p_i| \le \delta$, then $f_i = 1$, otherwise $f_i = 0$ for $i \in \{1..m\}$.
2. Also we need compute "Gamma Table" GT. Set GT[e] = f, where e denotes a symbol in the alphabet and $f = f_1..f_m$ is a binary word, if $|e - p_i| \le \delta$, then $f_i = |e - p_i|$, otherwise $f_i = 0$ for $i \in \{1..m\}$. Here, each f_i is stored as a binary number of *d* bits where
3. Let "LS" is a bit-wise operation that shifts the bits of a binary word by one position to the left. The definition is as follows:

   $DState_r = (LS(DState_{r-1}) \text{ OR } 1) \text{ AND } DT[S_r]$ (1-4)

   For r = 1..m and initial $DState_0 = 0$
4. Let "RS" is a bit-wise operation that shifts the bits of a binary word by *d* position to the right. The definition is as follows:

   $GState_r = RS(GState_{r-1}, d) + GT[S_r]$ (1-5)

   For r = 1..m and initial $GState_0 = 0$
5. If and only if the m-th bit of $DState_r$ is 1 or equivalently if and only if $DState_r$ is greater than or equal to 2^{m-1} when it is viewed as a decimal integer, and the m-th block of *d* bits taken as an integer is less or equal to γ, then we say that there is (δ, γ)-approximate string matching at position (r-m+ 1).

3.3 (δ, γ)-Approximate String Matching Algorithm

The algorithm is a union of both δ-approximate string matching and γ-approximate string matching. Namely, if and only if condition (1-4) and (1-5) is satisfied at the same time, then that is (δ, γ)-Approximate String Matching Algorithm.

Example: In example, let Σ= {1,2..9}. We suppose P= {3,4,6,2}, S={3,4,6,2,8,2,4,5,7,1}, δ=1,γ=3. In the preprocessing table, DT[e] denotes the position where $|e - p_i| \le \delta$. For example, DT[4] = 0011 because $|4 - p_i| \le 1$ for i = 1,2. And also we will use blocks of size three (*d* = 3) to store the $|e - p_i|$ values where $|e - p_i| \le \delta$. For example, GT[3] = 000 001 000 001 because $|4 - p_i| \le 1$, for i = 1,2,4 (see Table 1,2,3,4)

Here note that we find two matchings for our query in Table 2. Their end of position is 4 and 10 respectively. In fact, this is just retrieval by the single δ-approximate string matching that we mentioned above. Then, as shown in Table 4, using the proposed matching algorithm, we only find one matching, that is, we did not find the second result that it cannot be inappropriate for user query. Because at position 10, it is 100, that is 4>γ= 3. Thus, its efficiency is improved.

A complete specification of the algorithm is listed as follows:

[Algorithm 1]

Procedure Shift (P, S, δ, γ) $\{n = |S|,\ m = |P|\}$

1. ***Begin***
2. $DT_i[e] = \begin{cases} 1 & if\ |e - p_i| \le \delta \\ 0 & otherwise \end{cases} \qquad \forall_i \in \{1..m\}, \forall_e \in \Sigma$
3. $GT_{d..d..d-1}[e] = \begin{cases} |e - p^i| & if\ DT_i[e] = 1 \\ 0 & \end{cases} \qquad \forall^i \in \{1..m\}, \forall^e \in \Sigma$
4. $DState_0 \leftarrow 0$
5. $GState_0 \leftarrow 0$
6. ***For*** r ← 1 to n ***do***
7. $DState_r \leftarrow (LS(DState_{r-1}) \text{ OR } 1) \text{ AND } DT[S_r]$
8. $GState_r \leftarrow RS(GState_{r-1}, d) + GT[S_r]$
9. ***If*** $[DState_r]_{10} \ge 2m-1 \text{ AND } GState_{dm-d..dm-1} \le \gamma$ ***Then*** write (r-m+ 1)
10. ***End For***
11. ***End***

[End of Algorithm 1]

Table 1. "Delta Table" DT

i	P_i	DT[1]	DT[2]	DT[3]	DT[4]	DT[5]	DT[6]	DT[7]	DT[8]	DT[9]
4	2	1	1	1	0	0	0	0	0	0
3	6	0	0	0	0	1	1	1	0	0
2	4	0	0	1	1	1	0	0	0	0
1	3	0	1	1	1	0	0	0	0	0

Table 2. Computing DStates and Finding δ-approximate matching

r	1	2	3	4	5	6	7	8	9	10
S_r	3	4	6	2	8	2	4	5	7	1
$LS(DState_{r-1}) OR T[S_r]$	0001	0011	0111	1001	0011	0001	0011	0111	1101	1001
$DT[S_r]$	1011	0011	0100	1001	0000	1001	0011	0110	0100	1000
$DState_r$	0001	0011	0100	1001	0000	0001	0011	0110	0100	1000
$[DState_r]_{10}$	1	3	4	9	0	1	3	6	4	8

Table 3. "Gamma Table" GT

i	P_i	1	2	3	4	5	6	7	8	9
1	3	000	001	000	001	000	000	000	000	000
2	4	000	000	001	000	001	000	000	000	000
3	6	000	000	000	000	001	000	001	000	000
4	2	001	000	001	000	000	000	000	000	000

Table 4. Finding (δ, γ)-Approximate String Matching

r	1	2	3	4	5	6	7	8	9	10
S_r	3	4	6	2	8	2	4	5	7	1
3	000	001	000	001	000	001	001	000	000	000
4	001	000	001	000	001	000	001	010	000	000
6	000	001	000	001	000	001	000	010	011	000
2	001	000	001	000	001	000	001	000	010	100

4. PERFORMANCE EVALUATION

For testing the efficiency of the proposed algorithm, a series of experiments are performed to compare with the related approach. In these experiments, we suppose these melodies have been represented as a string of pitch values, then we use the algorithm proposed to search the string queried from music database.

In the experiment, queries are performed in a sample musical collection and they have been represented as a string of pitch values (see Figure 1), and then assume two patterns to retrieve. For first pattern, we use δ=1,2,3,4,5 and γ=1,2,3 respectively to perform by δ-approximate string matching and (δ, γ)-approximate string matching. Also for the second pattern, we use δ=1,2,3,4,5 and γ=1,2,3,4 respectively.

Here, we propose a variable named Precision-Matching (PM) that is the exact rate of matching query. Thus, we use variant δ, γ values to perform respectively, and then we compared the two approximate string matching (see Figure 2) by PM. The experimental result shows that (δ, γ)-approximate string matching algorithm is more efficient than the single δ-approximate string matching. Furthermore, their average PM is compared, and the result shows again that the (δ, γ)-approximate string-matching algorithm has higher average PM than δ-approximate string matching (see Figure 3)

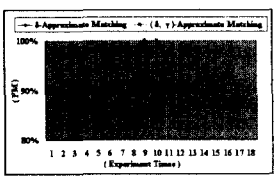


Figure 2. Comparison of PM in Two Approximate Matching

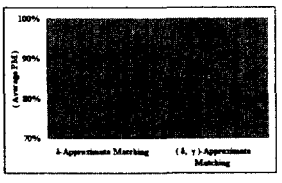
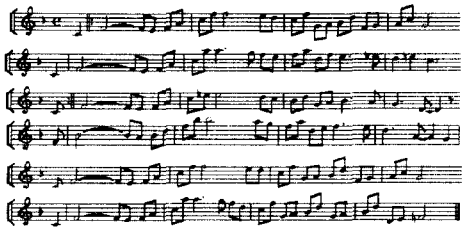


Figure 3. Comparison of Average PM



The two Pitch Value Pattern: {5,-1,1,4,3,5,0} and {-1,-2-2}

The Pitch Value String is: {5,-1,1,4,3,5,0,-1,-2,-2,5,-10,2,1,4,-9,2,2,3,-5,-7,5,-1,1,4,3,9,0,-2,-2,-1,1,4,-7,3,-1,-1,-1,2,-4,-12,5,-1,1,4,3,3,0,-1,-2,-2,4,-7,2,1,-1,-2,-5,3,5,-1,1,4,3,5,0,-1,-2,-2,4,-7,2,1,-1,-2,-5,-2,-7,5,-1,1,4,3,5,0,-1,-2,-2,5,-10,2,1,4,-9,2,2,3,-5,-7,5,-1,1,4,3,9,0,-2,-2,-3,-2,5,-10,2,1,4,-7,2,1,4,-12,2,1}

Figure 1. Melodies of Schumann's Träumerei

5. CONCLUSION

Music is an important type of media. It is necessary to retrieve for music database, which is just many users' requirement.

In this paper, we proposed an efficient algorithm using melodies represented as a string of pitch interval for musical retrieval. Here, (δ, γ)-approximate string matching algorithm refers to the process of looking for occurrences of a string over a given alphabet within a set of sequences over the same alphabet. The sequential pattern is the result of transcribing the humming or sung query while the sequence set is the music database.

Through analyzing the experimental results, we can know that union of two approximate string matching, (δ, γ)-approximate string matching algorithm is better and more efficient than single δ-approximate string matching algorithm.

In the future research, it is necessary to extend the proposed algorithm to consider more general situation.

REFERENCE

[1] Ghias A., Logan H., C.D., "Query by Humming: Musical Information Retrieval in an Audio Database," *In Proceedings of Third ACM International Conference on Multimedia*, 1995.

[2] E. Cambouropoulos, M. Crochemore, C.S. Iliopoulos, L. Mouchard, "Algorithms for Computing Approximate Repetitions in Musical Sequences," *In Proceedings of the 10th Australasian workshop*, 1999.

[3] E.C. and G. Widmer, (2000a) "Melodic Clustering: Movtivic Analysis of Schumann's Träumerei," *In Proceedings of the III Journeyed Information Musicale*, Bordeaux, France.

[4] Jia-Lian H., C.C. Liu, and Arbee L.P. Chen, "Efficient Repeating Pattern Finding in Music Databases," *In Proceedings of ACM Seventh International Conference.*

[5] Patrick A. V. HALL, "Approximate String Matching," *in Computing Surveys*, Vol. 12, No. 4, December 1980.

[6] Alexandra U., and Justin Z., "Melodic Matching Techniques for Large Music Databases," *In Proceedings of the International ACM multimedia*, 1999.

[7] A. L. Uitdenbogerd and J. Zobel, "Manipulation of Music For Melody Matching," *In Proceedings of the international ACM multimedia*, 1998.