Multimed Tools Appl (2010) 46:1–23
DOI 10.1007/s11042-009-0303-y

Algorithmic compositions based on discovered musical patterns

Man-Kwan Shan · Shih-Chuan Chiu

Published online: 26 May 2009

Abstract Computer music composition is the dream of computer music researchers. In this paper, a top-down approach is investigated to discover the rules of musical composition from given music objects and to create a new music object of which style is similar to the given music objects based on the discovered composition rules. The proposed approach utilizes the data mining techniques in order to discover the styled rules of music composition characterized by music structures, melody styles and motifs. A new music object is generated based on the discovered rules. To measure the effectiveness of the proposed approach in computer music composition, a method similar to the Turing test was adopted to test the differences between the machine-generated and human-composed music. Experimental results show that it is hard to distinguish between them. The other experiment showed that the style of generated music is similar to that of the given music objects.

Keywords Algorithmic composition · Data mining · Music style · Repeating patterns

1 Introduction

Computer music generation is the dream of computer music researchers. The process of producing musical content consists of two major steps: composition and arrangement. First, composers create original melodies with chords in the basic structure. Next, arrangers rewrite and adapt the original melodies and chords specifying harmonies, instrumentation, style, dynamics, and sequence. After these two steps, performance, recording, mixing, and audio mastering are conducted to produce the music for the general listener. Composing needs most originality and shows the most valuable part among these processes.

Part of the content of this paper has been published in IEEE Proceedings of International Conference on Systems, Man, and Cybernetics, 2006.

M.-K. Shan (✉) · S.-C. Chiu
Department of Computer Science, National Chengchi University, Taipei, Taiwan
e-mail: mkshan@cs.nccu.edu.tw

S.-C. Chiu
e-mail: scchiu@cs.nctu.edu.tw

Machine-generated music compositions have been investigated for a long time. Current research on computer composition may be classified under two approaches based upon the method used to generate the compositional rules. In the explicit approach, the compositional rule is specified by humans while in the implicit approach the composition rule is derived from sample music. Therefore, in the implicit approach, training data is required in order to discover the compositional rules.

As shown in Fig. 1, there are four design issues regarding the implicit approach: feature extraction, feature analysis, rule learning and music generation. Feature extraction concerns the extraction of low-level music features from sample music. Feature analysis obtains the high-level semantic information from low-level music features. Rule learning discovers the patterns (compositional rules) in terms of the high level semantic information from the set of sample music. Finally, music generation employs the discovered patterns to generate music.

A musical work, the result of a composer's skillful training, is expected to possess rich compositional information. In a set of musical works from whether composed by the same composer or collected from an audience's personal preference, this set of musical works must share the commonly compositional patterns. From the rich background and nature of extant musical works, this study seeks to discover more about their compositional rules. Data mining technique is applied since it has been the adaptive ability to discover variant patterns.

In this paper, the implicit approach to computer music composition based on the discovered musical patterns from music examples is investigated. The developed approach takes a set of user-specified music examples as input and generates music with a style similar to that of the user-specified music set. While most of previous works on algorithmic music composition adopt the *bottom-up* approach, this paper develops a *top-down* approach that composers might prefer for music composition. A new framework considering the properties of music structures, melody styles and motif development is proposed. Furthermore, most of previous works characterize the melody style as repetition or statistics of notes which are in the level of musical surfaces, the proposed approach models melody styles in terms of melody features hidden from the melody surfaces. More specifically, the proposed approach is developed based on the data mining techniques in order to discover music style patterns. With respect to the algorithmic compositional

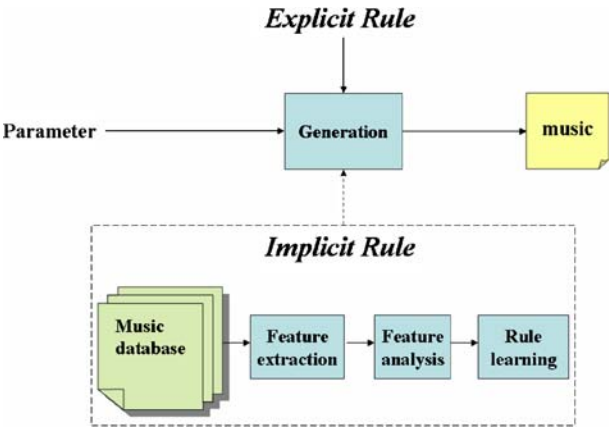

Fig. 1 Flow chart of computer-generated music composition

models, the proposed approach is a hybrid model with a learning procedure. Finally, we examine proposed system by a critical approach which is similar to Turing-test.

The remainder of this paper is organized as follows. In the next section, there is a review of previous work on computer music composition. Section 3 provides the system architecture and feature extraction of the proposed approach. Feature analysis and rule learning are described in Section 4. Section 5 presents the method of music generation. Next, in Section 6, the experiments are presented. At last, the conclusion is drawn in Section 7.

2 Related work

There exist many systems for computer music composition. Basically, there are two different types of computer music composition systems. One is the *algorithmic composition* which generates music automatically while the other is the *computer-aided composition* which servers as a tool to help the composer to generate music. Although the first commonly recognized algorithmic composition system pioneered in the mid 1950s, there is no universal classification for different approaches of algorithmic composition. In this section, we review approaches of algorithmic composition according to the method used to obtain the composition rules as mentioned in Section 1.

2.1 Composing with explicit rules

Early work on the generation of computer music focuses on compositions with rules explicitly specified by composers.

Stochastic system generates the sequence of musical parameters by selection from a given set according to specified probability distribution function. This random process fulfils the expectation of musicians. Therefore, the compositional rule is embedded within the stochastic process itself. Hidden Markov Model (HMM) is probably the most common approach of stochastic process for music composition. Especially, in Markov model, the probability of future events depends on past events. This fulfils the sequence requirement of melody. An example is CAMUS which consults a user-specified Markov table to control rhythm and the temporal organization of note group [32].

The hierarchical structure of musical thought has a strong similarity to linguistics. While formal grammars and finite automata are normally utilized for representation of the laws of formation of a language, some music composition mechanisms are developed based on formal grammars or automata. BP2 is an example of software developed for composing using grammars. Production rules whose terminal nodes correspond to sound objects are employed to generate music with constraint-satisfaction programming [4].

Artificial life is a discipline that studies natural living systems by computer simulation of their biological aspects. Cellular automata and genetic algorithms (GA) are two common approaches for algorithmic music composition. A cellar automata consists of an array of cells that change states according to evolution rules. While cellar automata generates tremendous amount of patterns, music composition can be modeled as a pattern propagation process. CAMUS, which consults Markov chains to generate rhythm, is a cellar automata music generator. It utilizes both Game of Life and Demon Cyclic Space automation to produce triple of notes and to determine the orchestration, respectively, in parallel [30].

Music composition can be treated as a combinatorial optimization problem. Therefore, it is natural to develop music composition mechanism based on genetic algorithm. In GA,

initially, a population of entities is randomly created. Reproduction which involves mutation process is performed to create the next generation. This reproduction process repeats until the population passes the evaluation specified as the fitness function. In terms of music composition based on GA, musical phrases, with a higher, post-evaluation fitness value, are selected for creating the next generation. The desired musical phrase is generated gradually after each round. Therefore, the fitness function acts as the compositional rules. One GA-based example is offered by J. A. Biles who proposed the GenJam system that generates jazz solos [5]. Another was designed by G. Papadopoulos et al. [35] who utilized the genetic algorithm to generate jazz melody. The major difference between these two approaches lies in the fitness function. The fitness function of the former is evaluated interactively by human while that of the latter is evaluated against eight characteristics of the melody. More discussions on GA-based music composition can be found in [7, 50].

The particle swarm system, SWARMUSIC [6], employs particle motion to produce musical material by mapping particle position onto a MIDI event. This system can also improvise and interact with users in real-time. Similar idea is employed to generate music by simulating moves of artificial ants on a graph where vertices represent notes and edges represent possible transitions between notes [23]. Another interesting approach is developed recently in the field of artificial chemistry. To compose music using the virtual model of the chemical system, molecules and chemical equations are designed to implement a set of musical rules. The implemented system based on a non-deterministic algorithm successfully generates musical phrases [33, 44].

2.2 Composing with implicit rules

Recent work on computer-generated musical composition has been attempted to learn the features or rules of composition from examples of musical work implicitly.

Some approach comes with the development of the machine learning technology. Reagan [38] adopted decision tree classifier to discover the common properties of the meta-level features from the examples labeled as scores and to composes music using these features. At the IRCAM research center, S. Dubnov et al. proposed a machine learning methods for musical style learning from examples [20, 28]. Two different approaches, incremental parsing and prediction suffix tree are utilized to model patterns and to inference new music. Both approaches try to model musical repeating patterns in an approximate way.

D. Conklin [12] proposed the use of statistical model to capture music styles from a corpus of music work. Several methods for sampling from a statistical model along with the pattern-based sampling are investigated. Hidden Markov Model has been applied with success to implicit music composition. Cybernectic Composer is an earlier example based on HMM to generate music in different styles [3]. Farbood and Schoner present an algorithmic composition system which generates Palestrina-style music [22]. Palestrina was a famous sixteenth-century composer whose work is seen as a summation of Renaissance polyphony. Given counterpoint examples, the system infers the probability tables of Markov model and composes a counterpoint line given a cantus firmus. In most HMM-based approaches, each state represents a note. Recent approach treats each state as the patterns extracted from examples [48]. Therefore, this approach learns the composition rules from examples implicitly using sequences of patterns.

The artificial neural network (ANN) has been used in the last years for music composition. It resembles the activities of human brain structure. In this computation, the process is performed by a set of several simple units, the neurons, connected in a network

by weights. The parameters of the network are adjusted by learning from examples. The most recent research is conducted by Correa et al. Both supervised and unsupervised approaches were proposed to learn aspects of music structure and to compose new melody, by using Back-Propagation-Through-Time and Self Organizing Maps neural networks. [10, 17].

D. Cope [13–16] has conducted a series of work in algorithmic composition, whose basic idea is from Mozart's dice game. The basic approach is to extract signatures which are short musical passages from existing works by pattern matching techniques first. Then the extracted signatures are replicated and recombined to create a reasonable melody by evaluation against augment transition network borrowed from natural language processing. The composition process also takes other musical properties into consideration, such as phrase structures, texture, earmarks and unifications. The music produced by this kind of approaches has demonstrated an elegant result.

2.3 Discussions

All of the above-mentioned researches belong to the bottom-up approach where smaller segments are extended to compose higher level musical sections for the entire work. Furthermore, most approaches focus on learning patterns of note sequences while music structures and motives are not considered. In the majority of previous approaches, no special attention has been paid to the beginning and ending of generated melody, to say nothing of music structures. Motif is an important element in music composition. In the bottom-up approach, simple notes instead of motives are used as the basic units of melody. The top-down approach, on the other hand, starts by developing a composition plan and proceeds to refine this plan. This approach forces the composition to be creative within formal constraints, such as musical structure. While most, if not all, of the bottom-up approaches, deal with algorithmic composition as a problem solving task, the top-down approaches treat it as a creative process. Composers might prefer the top-down approach that helps the composer to be creative within the formal constraints of generative process. Compared with the above-mentioned bottom approaches, our proposed framework is a top-down approach that analyzes and learns the musical structure style, discovers the motives and the melody style from examples first. Then a new music object is generated by settling on the musical structure, determining the melody style and generating melody sequence.

3 System architecture and main melody extraction

Figure 2 shows the system architecture and the process flow of the proposed music system. Given a set of music in MIDI format, the main melody extraction component extracts the main melody which is a monophonic melody. Then, each extracted melody is analyzed using the music structure analysis component, the melody style analysis component and the motif mining component individually. All analysis steps are off-line processes, namely, all music objects in database can be analyzed before using system. In contrast, the learning and the generation procedures are performed after the music objects are specified by the user. The structure analysis component detects the musical section-phrase structure, extracts section features and generates the section sequences for the structure learning component. The structure learning component generates the music structure model by statistics analysis of section and phrase features. The melody style analysis component extracts the melody style by finding accompanied chords and generates different melody style representations for the melody style learning component. The melody style learning component discovers

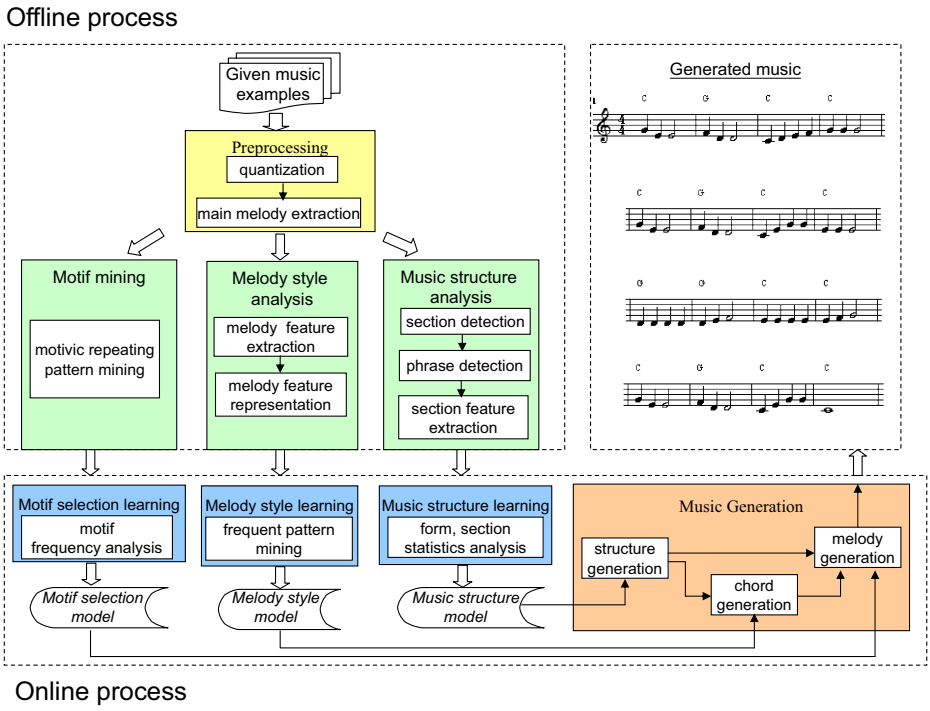

Fig. 2 The system architecture and process flow of the proposed approach

the melody style model by frequent pattern mining techniques. The motif mining component finds the set of motives which constitute the potential candidates by motivic repeating pattern mining technique for the motif selection learning component. The motif selection learning component generates the motif selection model by frequency analysis of extracted motives. Finally, after these analysis and learning processes, the three models (music structure model, melody style model and motif selection model) are established. These three models involve the music style of the selected music objects. In the music generation component, a new music object is generated based on these three models.

Melody is the essential element of music composition. The main melody extraction component consists of two steps. In the first step, quantization is performed. In digital music processing, quantization is the process to align music notes to a precise setting. This is achieved by setting the timing grid and moving each note to the closest point on the timing grid. In our system the timing grid is set to a quarter of a beat and quantization is applied to both MIDI's Note-On messages and Note-Off messages. Figure 3 illustrate an example of quantization process.

The next step extracts the monophonic melody from the polyphonic melody. Uitdenbogerd et al. [45] have presented four melody extraction methods: All-mono, Entropy-channel, Entropy-part and Top-channel. According to their experimental result, All-mono method, adopted in this paper, obtains the best accuracy. All-mono combines all the music tracks and among the simultaneous notes, includes the highest notes as the main melody. In Fig. 4, the first staff (Fig. 4a) shows an example of polyphonic segment while the second staff (Fig. 4b) shows the extracted main melody using All-mono.

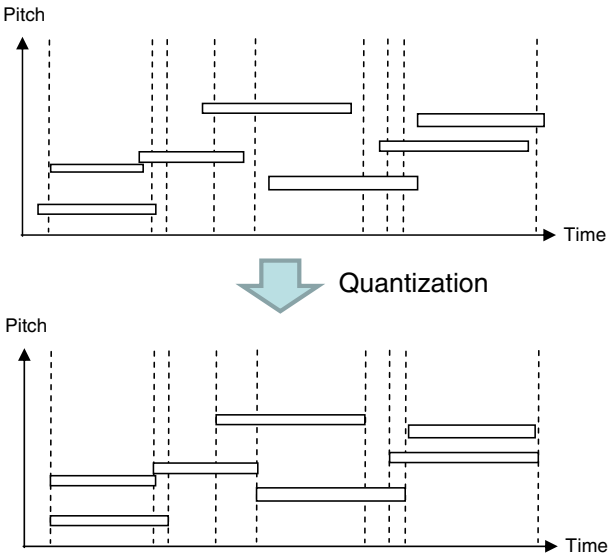

Fig. 3 An illustration of quantization

4 Analysis and learning

4.1 Music structure analysis and structure rule learning

Musical structure can be regarded as a hierarchical structure similar to the structure of an article. In our approach, a music object is composed of sections and a section is composed of one or more phrases [19]. The structure analysis component discovers the section-phrase hierarchical structure of a music object while the structure learning component analyzes common characteristics from structures of given music examples. Some researchers have devoted to the investigation of music segmentation techniques for symbolic music [8, 11, 31, 43, 49]. Most work focuses on the extraction of phrases. Takasu et al. proposed the phrase extraction method based on some heuristic rules and extended the work to classify phrases into two classes: theme phrases and non-theme phrases based on the decision tree algorithm [43]. Chen et al. adopted a similar approach to extract phrases and to group similar phrases for sentence extraction [11]. Nevertheless, to the best of our knowledge, nobody has focused on the detection of musical sections which constitute the musical form. Therefore, we present a heuristic method for section detection.

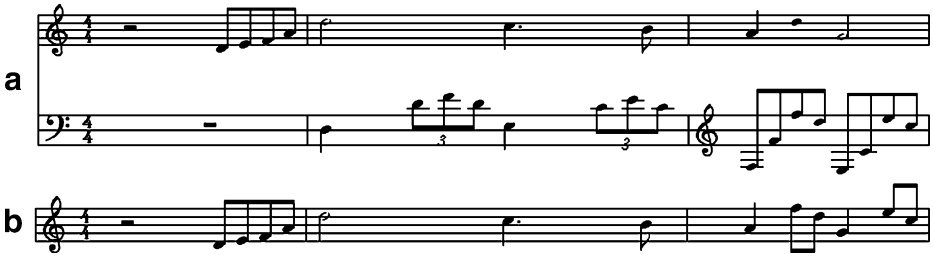

Fig. 4 An example of All-mono method to extract the main melody

There are three main steps in the music structure analysis component. In the first step, section detection is performed. To find the section structure in music, first we discover repeating patterns over main melody. A repeating pattern is a sequence of notes repeats several times in a music object. The main melody is a note sequence where a note can be parameterized with several property values such as pitch, duration and velocity. Velocity is only relevant to music performance. Therefore only pitch and duration are considered for the structure analysis. The repeating patterns over pitch-duration sequence are discovered by repeating pattern mining techniques. Over the past few years, several algorithms have been proposed to mine repeating patterns in a sequence [25, 27, 29, 46]. Suffix tree is a well-know data structure originally developed for string matching. It is also utilized to find the repeating patterns by storing the suffix information [29]. There may exist a large number of repeating patterns in a sequence. Therefore, the concept of non-trivial repeating patterns was introduced by Hsu et al. [26]. A repeating pattern is non-trivial if and only if there does not exist another superstring with the same frequency. Hsu et al. proposed two different approaches, correlative matrix and string-join, to mine non-trivial repeating patterns [25]. In the former approach, a data structure called correlative matrix is constructed by lining up the note sequence the horizontal and vertical dimensions respectively to keep the intermediate information of substring matching. Each cell of this matrix denotes the length of a founded repeating pattern. After the construction of this matrix, the repeating frequencies of all repeating patterns can be derived by computing the non-zero-cells in the matrix. The latter approach, string-join, utilizes the anti-monotony property to avoid generating large amount of candidates. In this approach, shorter patterns are joined into longer ones and the non-qualified candidates are pruned out. Here, we employ the correlative matrix technique to find the repeating patterns.

After the repeating pattern mining process, a music object may contain more than one repeating pattern. Each repeating pattern appears several times. Figure 5 illustrates the instances of non-trivial repeating patterns after correlative matrix technique has been applied to the musical piece “Little Bee.” Repeating patterns with durations shorter than two bars are not shown here. In Fig. 5, each strip denotes an instance of a non-trivial repeating pattern. There are six non-trivial repeating patterns. For example, the first pattern NTRP0 has three instances. One starts at the first bar, another starts at the fifth bar, and the other starts at the thirteenth bar.

As not all instances of repeating pattern are required for section structure detection, appropriate instances need to be selected. First, in our proposed approach, all the instances of the repeating patterns with durations shorter than two bars are filtered out. Then we attempt to extract the musical sections by finding the set of non-overlapping repeating pattern instances such that the total length of the selected instances is maximized.

We modify the algorithm for exon-chaining problem developed in the field of bioinformatics [26]. Given a set of weighted intervals in a chain, the exon-chaining problem attempts to find a set of non-overlapping intervals such that the total weight is maximal. This algorithm can be modified to accommodate the pattern selection problem by replacing the weight of an interval with its duration. As the detailed pattern-selection algorithm in Fig. 6 shows, given n pattern instances, this problem can be solved using dynamic programming in a one dimension array S of $2n$ elements, n of which corresponds to the starting (left) positions of the instances and n of which corresponds to the ending (right) positions of the phrases. For the sake of simplicity, it is assumed that all instances are distinct. This algorithm starts by sorting the starting and ending positions of all instances

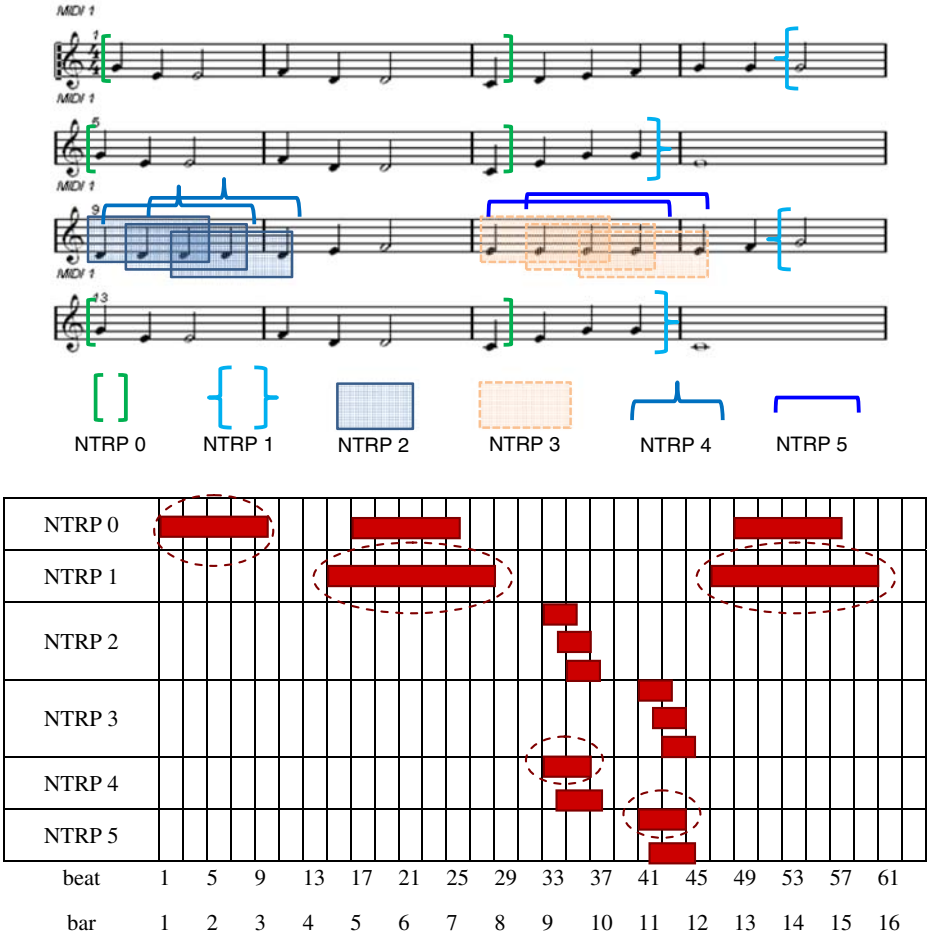

Fig. 5 The instances of repeating patterns in the main melody of "Little Bee"

into a one dimension array T of $2n$ elements. The i-th element of S, S_i, represents the maximum duration for the set of instances which ends before the position T_i. The set of instances, which ends before or at T_i, to maximize the total duration either excludes the instance ends at T_i, or includes the instance ends at T_i to the maximum set of phrases which ends before the left position of this instance (Fig. 6, lines 5 to 11). This leads to the following formula

$$\begin{cases} S_i = \max\{S_{i-1}, S_{f-1} + T_i - T_f\}, & \text{if } T_i \text{ is the ending position of an instance starting at } T_f \\ S_i = S_{i-1}, & \text{, otherwise.} \end{cases}$$

The set of selected phrases can be derived by backtracking the array S (Fig. 6, lines 13 to 20). To describe the computation of this algorithm, Fig. 7 gives the example corresponding to the pattern instances in "Little Bee." In this example, the duration of each instance is measured in beats. The circled instances constitute the set which maximizes the total duration.

Algorithm Pattern-Selection
Input: A set of n phrase, $P = \{p_i = (l_i, r_i) \mid 1 \le i \le n\}$
Output: The subset Q of P such that the total duration is maximized
1) Sort the left positions and the right positions of the phrases into a 1-dimension array T of $2n$ elements
2) Create a 1-dimension array S of $2n$ elements
3) **for** $i \leftarrow 1$ **to** $2n$ //initializing value for S
4) $S_i \leftarrow 0$
5) **for** $i \leftarrow 2$ **to** $2n$ //fill up array in order
6) **if** $T(i)$ is the right end of a phrase $p_k = (l_k, r_k)$
7) $f \leftarrow l_k$
8) $d \leftarrow r_k - l_k$
9) $S_i \leftarrow \max\{S_{f-1} + d, S_{i-1}\}$
10) **else**
11) $S_i \leftarrow S_{i-1}$
12) $i \leftarrow 2n$; $Q \leftarrow \{\}$;
13) **While** $(i \ge 2)$ //backtracking s_{2n} to extract phrases
14) **if** ($S_i \neq S_{i-1}$)
15) let p_k be the phrase ends at T_i
16) Add p_k to Q
17) $i \leftarrow l_k$
18) **else**
19) $i \leftarrow i$-1
20) **return** Q

Fig. 6 Pattern-selection algorithm

The next step of section structure detection is to find the section boundary based on the selected pattern instances. Each selected instance corresponds to a section. Initially, the starting and ending positions of a section are set to those of corresponding instance. Then each section is shrunk or expanded by adjusting the starting position to the nearest starting position of an odd numbered bar. For example, the second section in Fig. 8 is shrunk by

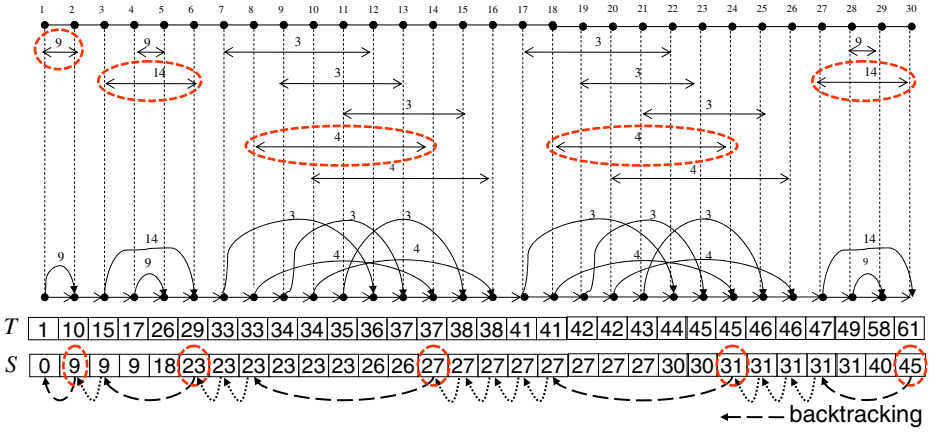

Fig. 7 An illustration of example of pattern selection corresponding to Fig. 5

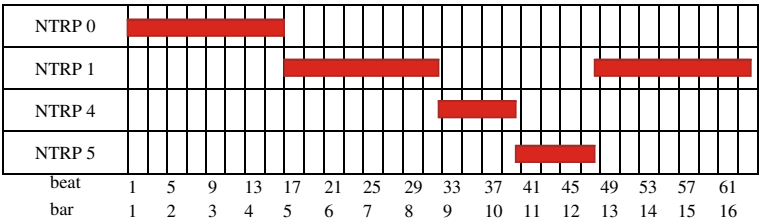

Fig. 8 The selected pattern instances of the example in Fig. 5

adjusting its starting position to the first beat of the fifth bar. At last, each section is expanded by aligning its ending position to meet the starting position of the next section. For example, in Fig. 8, the ending position of the first section is set to the end of the fourth bar.

The second step in music structure is phrase detection. To obtain the number of phrases in a musical section, the LBDM (Local Boundary Detection Model) approach developed by Cambouropoulos et al. [8] is used to segment a section into phrases. Previous experiments have adopted LBDM as representative of the melodic feature-based algorithms for melody segmentation [31]. LBDM extracts the pitch interval sequence, the inter onset interval sequence and the reset sequence from the main melody. Then these three sequences are integrated into the sequence of boundary strength values measured by the change rule and the proximity rule. The resulting peaks of the boundary strength value sequence are regarded as the phrase boundaries.

The final step in music structure analysis is to extract features for each section, including section labeling. Each section is labeled such that all the instances of the same repeating pattern are labeled with the same symbol. For example, in Fig. 8, the labeled sequence becomes ABCDB. The second and the fifth section correspond to the same repeating pattern, so do the third and the fourth section. At last, the structure analysis component outputs a section sequence where the section is parameterized by *label*, *numberOfOccurrences*, *numOfPhrases* and *length*. While the attribute *label* denotes which label it is, the attribute *numberOfOccurrences* denotes the number of appearances of the same label. The attribute *numOfPhrases* denotes the number of phrases in this section and the attribute *length* denotes the length of the section measured in beats.

In the learning step, the probability distribution of section sequences along with associated meters is derived to model the style of musical form. Moreover, the conditional probability Prob(*numOfPhrases*|*label*, *numberOfOccurrences*) of the number of phrases given a label and an occurrence value is derived. So does the conditional probability that Prob(*length*|*label*, *numberOfOccurrences*) of the section length given a label and an occurrence value.

4.2 Melody style analysis and melody style rule learning

After the analysis of the musical structure, the melodies are segmented into sections. The segmented melodies are collected for the purpose of melody style mining. The design issues regarding melody style mining techniques are melody feature extraction, melody feature representation and melody style mining/classification algorithms. In a sense, most works on algorithmic music composition may be regarded as approaches to model melody styles and

to generate note sequences based on the models. For example, the HMM-based approach models melody style as Hidden Markov Model where the probability of next note depends on one or more previous notes. In addition, there exist some researches on melody style analysis from symbolic music [9, 18, 37, 39, 40, 47]. The work developed in MIT Media Lab. [9] employed Hidden Markov Model to model and classify the melodies of Irish, German and Austrian folk song. Melodies are represented as a sequence of absolute pitches, absolute pitches with duration, intervals and contours. Another research in CMU utilized the naïve classifier, linear and neural network respectively to recognize music style for interactive performance systems [18]. Thirteen statistical features derived from MIDI are identified for learning of music style. Kranenburg and Backer extracted 20 low level characteristics of counterpoint for polyphonic music style recognition. The K-means clustering, the k-nearest-neighbor and C4.5 classification algorithms are employed to obtain music styles [47]. In [37], 28 statistics of melody content, such as number of notes, pitch range, average of note duration, number of diatonic notes, are developed for feature representations of automatic music style recognition from symbolic representation of melodies. The Bayesian classifier, the k-nearest neighbor and the self-organizing map are applied to perform music style learning. Most of these approaches characterize the melody style as repetition of notes or statistics of pitches which are in the level of musical surfaces. However, stylistic features that characterize the music are usually hidden from the melody surface.

We have proposed the music style mining technique to construct the melody style model for a set of music objects [39, 40]. The basic idea is to utilize the chord based on harmony to extract the melody features. A chord is comprised of a number of pitches sounded simultaneously. Accompanying chords can be used as the melody feature to characterize the melody style. Figure 9 illustrates the reason why we choose chord as the feature for melody style mining. Figure 9a and b show two different music segments. These two segments are quite similar in terms of either note sequence or interval contour. However, the music style and feelings of them are quite different. On the contrary, both music segments composed by Bach shown in Figure 9c and d are of the same style. The chords assigned to these two music segments are the same.

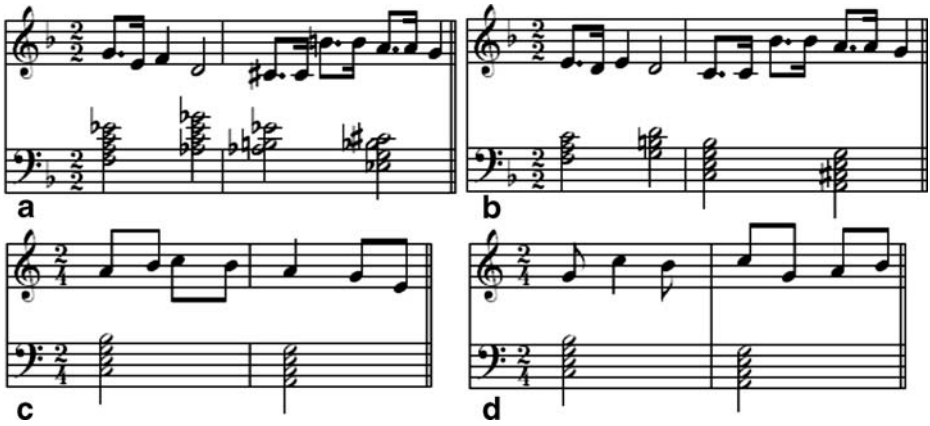

Fig. 9 Examples of music segments and assigned chords [40]

To match up the chords and melody, we have developed the chord assignment method based on the music theory. This algorithm starts by the determination of the chord sampling unit. Then, for each sampling unit, sixty common chords are selected as the candidates. Each candidate is scored based on harmony and chord progression rules. The chord with the highest score is assigned to the respective sampling unit. The detailed algorithm can be found in our previously published paper [39, 40].

After determining the chord respective to each sampling unit, the melody feature can be represented in the following different ways.

(1) Chord sequence: the feature of a melody is represented as a sequence of chords.
(2) Set of chord bi-grams: the feature of a melody is represented as a set of bi-grams of chords. A bi-gram is an adjacent pair of chords extracted from a chord sequence.
(3) Chord set: the feature of a melody is represented as a set of chords.

For instance, assume that in a melody of four sampling units, the chords with the highest scores are C, G, G, and C respectively. The melody feature is represented by the chord set {C, G}, the set of chord bi-grams {CG, GG, GC}, and the chord sequence <CGGC>.

In order to obtain the interesting hidden relationships between chords and music styles, we adopted two melody mining methods with respect to the melody feature representations.

Frequent itemset If the feature of a melody is represented as a set of chord or a set of chord bi-grams, the concept of frequent itemset mining in the association rule mining is utilized. In our work, each item corresponds to a chord or a chord bi-gram. An itemset is frequent if the number of music examples that contain this itemset is larger than a specified minimum support *minsup*. Assume that there exists a frequent itemset {C, F, G} from a set of lyric-style melody examples, this represents that a great part of lyric-style melody examples consist of chords C, F and G together. The Apriori algorithm is employed to find the frequent itemsets [1]. Apriori is a well-known data mining approach originally developed for the discovery of frequent itemsets from a database of itemsets. The classic Apriori algorithm for the discovery of frequent itemsets makes multiple passes over the database. In the first pass, the support of each individual item is calculated and those above the *minsup* are kept as a seed set. In the subsequent pass, the seed set is used to generate new potentially frequent itemsets, namely candidate itemsets. Then the support of each candidate itemset is calculated by scanning the database. The candidates with support no less than the *minsup* are the frequent itemsets and are fed into the seed set that will be used for the next pass. The process continues until no new frequent itemsets are found.

Frequent substring If the feature of a melody is represented as a sequence of chords, to find the ordered patterns, we mine the frequent substring, by modifying the concept of sequential pattern mining in sequence data mining techniques. The substring is consecutive, which differs from the sequential patterns. A substring is frequent if the number of music examples, which are the superstrings of this substring, is larger than a specified minimum support. The frequent substring is found by modifying the join step of the Apriori-based sequential mining algorithm [2].

Given a set of music examples, the discovered frequent chord patterns constitute the music style model of user specified examples. Figure 10 gives an example of melody style mining. In this example, there are four music sections which are transformed into chord sequences. Given the minimum support 50%, eighteen frequent chord patterns are discovered and constitute the music style model.

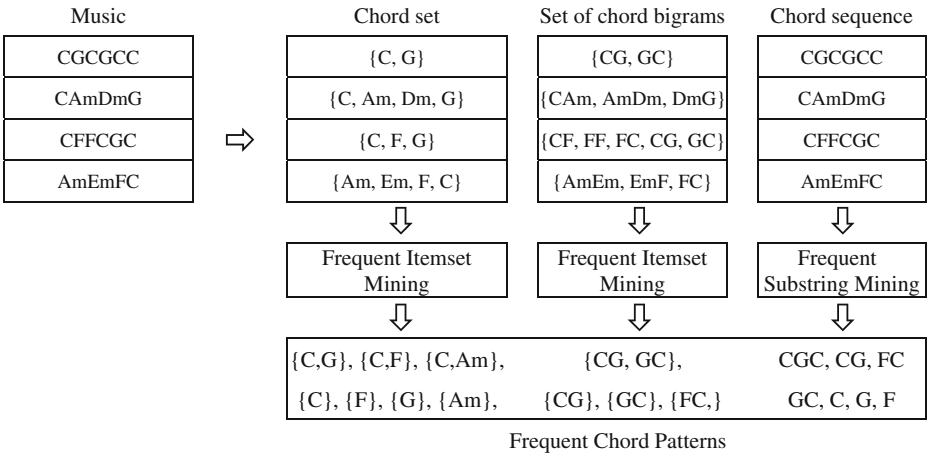

Fig. 10 An example of melody style mining

4.3 Motif mining and motif selection rule learning

In musicology, a motive is a salient recurring segment of notes that may be used to construct all or some of the melody and themes. Motif development is the compositional procedure in which complete work or sections are based a thematic motif [42]. There are several ways for developing a motif. Some of the major ways to develop a motif are repetition, sequence, contrary motion, retrograde, augmentation and diminution, as Fig. 11 shown. The following provides a brief description for each kind of motif developments.

(1) Repetition

The repetition is the exact repeating, where Fig. 11a is an example. Exact repetition is a motif development usually seen in composing and is usually used by composers to impress the audience.

(2) Sequence

The sequence refers to the moving of a motive in pitch in a constant level. It is a specialization of transposition. Figure 11b shows an example where the motive is moved downwards by two semitones.

(3) Contrary motion

Contrary motion performs notes within the original motif but in reverse contour while rhythm remains unchanged. More exactly, the contrary here indicates a pitch interval of two consecutive notes when the original motif is equal to the negative pitch interval of the two notes in derived motif. For example, in Fig. 11c, the interval sequence, in semitones, of the original motive is <2, 2, 1, −3, 2, −4> while that of the first variation is <−2, −2, −1, 3, −2, 4>.

(4) Retrogradation

To retrograde is to perform notes within the original motif in reverse order with the rhythm unchanged. As an example in Fig. 11d, the first measure is an original motif and the

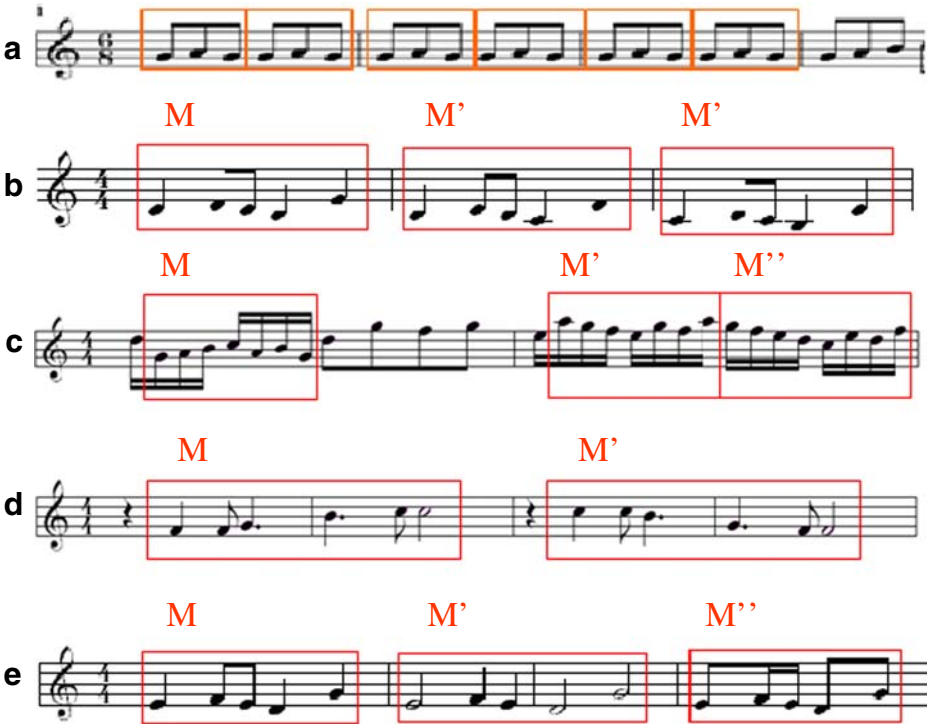

Fig. 11 Examples of the development of motif: **a** Repetition, **b** Sequence, **c** Contrary Motion, **d** Retrograde, **e** Augmentation and Diminution

second measure is the one that performs the retrograde variation. The rhythm of the derived motif isn't changed. However, the pitch sequence is varied from the original <65, 65, 67, 71, 72, 72> to <72, 72, 71, 67, 65, 65>.

(5) Augmentation or Diminution

Composers using augmentation or diminution motif development will keep the original melody with the rhythm scale enlarged (augmentation) or shrunk (diminution) proportionally. More precisely, augmentation involves an extended duration of each note in a motif while diminution involves a reduction in the duration of each note in a contrary motif. For example, in Fig. 11e, the first measure is an original motif, while the second and the third measures are the result of augmentation and diminution respectively.

To discover the motifs which are not necessarily exact repetition, the conditions of substring matching in each cell of the correlative matrix need to be modified in order to accommodate the motivic variations. For example, to discover the retrogradation, each cell of the correlative matrix should consult the lower-right cell, rather than the upper-left cell in exact repetition finding. More details concerning the motif finding algorithms can be found in earlier work completed by the authors [24].

The motif selection model describes the importance of motifs. Let $Freq_{m,music}$ denote the frequency of a motif *m* appearing in music object *music*. The formula is normalized as Eq. 1 and this is denoted as *Support(m, music)*. For a motif *m* in a given music database *DB*, its support are summed up and denoted as *ASupport(m, DB)*. Finally, the *ASupport* is

normalized as Eq. 3 and it is denoted by *NSupport(m,DB)*, where *Min(DB)* and *Max(DB)* represent the minimum and maximum *ASupport* of motives in *DB*, respectively.

$$Support(m, music) = Freq_{m,music} \Big/ \sum_{\forall motif \in music} Freq_{motif,music} \tag{1}$$

$$ASupport(m, DB) = \sum_{\forall am \in DB} Support(m, am) \tag{2}$$

$$NSupport(m, DB) = (ASupport(m, DB) - Min(DB) + 1)/(Max(DB) - Min(DB) + 1) \tag{3}$$

5 Music generation

In this section, the method used to generate music from the three models constructed in the previous steps is discussed. The flow chart is shown in the music generation component of Fig. 2, and Fig. 12 is an example of music generation.

According to the probabilistic distribution in the music structure model, the music generation component first generates the music structure expressed as a sequence of parametric sections, top level structure. As an example in Fig. 14, the structure model generates the sequence of the sections <(A,3,3,4), (A,3,3,4), (B,1,2,4), (A,3,3,4)>. For the description of each attribute, refer to the last paragraph in Sec. 4.2. According to these attributes, the system allocates phrases in each section which forms the second level. In general, the length of a phrase composed by musicians is a power of two bars [42]. Richard Strauss, describing his method of composing, has written "…*a motif or a melodic phrase of two to four bars occurs to me immediately. I put this down on paper and then expand it straightaway into a phrase of eight, sixteen, or thirty-two bars*… "

Therefore, given the length of a section and the required number of phrases, we design a heuristic algorithm, phrase allocation algorithm shown in Fig. 13, to allocate phrases. The basic idea of this algorithm is to allocate the phrases such that each is of length power of 2. As an example in Fig. 12, the length of the section A is four and the number of phrases is three. The algorithm allocates three phrases of length one, one, and two in order,

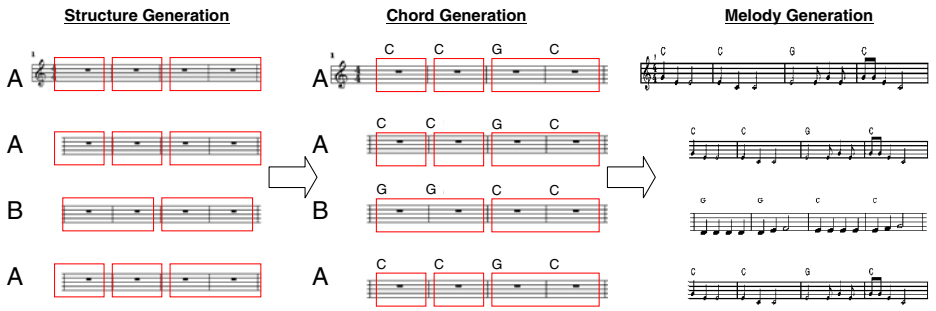

Fig. 12 An example of music generation

Algorithm Phrase-Allocation

Input: *length* (length of a section)

numberOfPhrases(number of phrases in this section)

Output: $\{phrase_i.length | 1 \le i \le numberOfPhrases\}$

1) $aveLength = length/numOfPhrases$

2) find x such that $aveLength - 2^x$ is minimum

3) for (i=1; $i \le numberOfPhrases - 1$; i++)

4) $phrase_i.length = 2^x$

5) $phrase_i.length = length - 2^x \times (i - 1)$

Fig. 13 Phrase-allocation algorithm

respectively. Note that if any label of the section is equal to 'A', it is applied similar arrangement of section A, directly.

After the determination of section-phrase structure, the chord generation component generates the chord for each bar based on the music style model. As stated in Section 4.2, the music style model consists of frequent chord patterns. The chord generation component randomly generates several chord sequences. The greater the number of frequent chord patterns contained in a randomly generated chord sequence, the higher the score of the chord sequence. The chord sequence with the highest score is assigned to the respective bar.

After the structure and chord information are determined, the melody generation component works as follows. For each phrase, the melody generation component selects a motif from the motif selection model. In general, the duration of a motif is shorter than that of a phrase. The selected motif is developed (repeated) based on the major ways of motif development mentioned in Section 4.3.

To ensure that the motif-developed sequence is harmonic to the determined chord sequence, an evaluation function is employed to measure the harmonization between a motif sequence and a chord sequence. This evaluation function is, in fact, the inverse function of the chord assignment algorithm mentioned in Section 4.2. In melody style

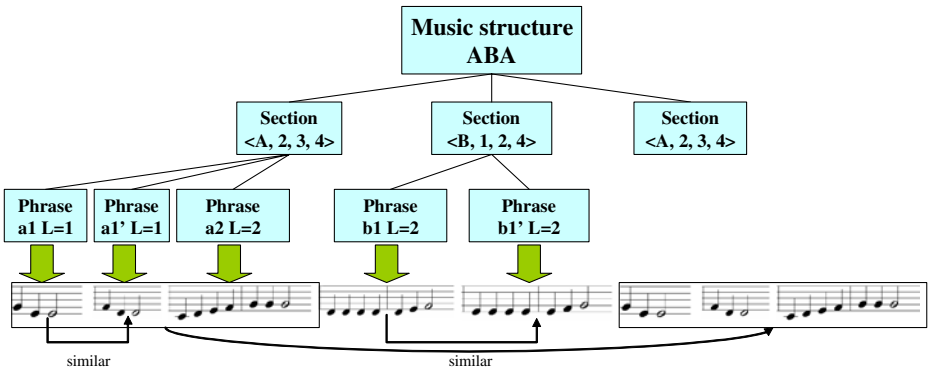

Fig. 14 An example of music structure generation and melody generation

mining, given a melody, the chord assignment algorithm tries to find the best accompanied chord sequence. In contrast, in melody generation, given a chord sequence, the evaluation function tries to find the best accompanied motif sequence. If the developed motif sequence is judged to be disharmonious, the melody generation component selects another motif from the motif selection model and develops the motif variation. This process is repeated until a harmonious motif sequence is produced.

Note that, from the perspective of music structure, some sections are associated with the same label. For instance, the example shown in Fig. 14 contains Sections 1 and 3 which are both associated with label "A." For those phrases contained in the repeated section, the motif sequences are simply duplicated from the motif sequences generated in the phrases of the previous section of the same label.

Finally, the melody generation component concatenates the motif sequences along with the corresponding chord sequences to compose the music.

6 Experiments

To demonstrate the performance of the proposed top down approach for algorithm music composition, we have implemented a music composition system on the World Wide Web (http://avatar.cs.nccu.edu.tw/~stevechiu/cms/experiment2). Our music composition system was implemented in Java along with jMusic [41] and Weka [51]. Both jMusic and weak are open source packages. jMusic is a Java library written for musicians. It is designed to assist the compositional process by providing an environment for musical exploration, musical analysis and computer music education. jMusic supports music data structure based upon note/sound events, and provides methods for organizing, manipulating and analyzing musical data. Weka is a collection of libraries for data mining tasks. It contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. In our implementation, jMusic is utilized to extract MIDI messages, maintain music data structure and output MIDI message. The chord assignment algorithm in the melody style analysis component is also developed in jMusic. Weka is used to implement the music style mining component.

Little attention in the research literature has been paid to the problem of evaluating the music generated by systems for algorithmic music composition. This comes from the fact that evaluation of aesthetic value in works of art often comes down to individual subjective opinion. The majority of music composition systems proposed in the research literature evaluate the performance by presenting the examples of composed music works only. Some researches performed qualitative analysis by asking subjects about the preference of the generated music works. Only few researches have conducted experiments for quantitative analysis of performance by asking subjects to discriminate system generated music from human composed music. However, to the best of our knowledge, there exists no research on algorithmic composition which performs comparative analysis of performances among different systems. This is partly owing to the availability or implementation of other music composition systems.

To evaluate the effectiveness and efficiency of the proposed music generation approach, three experiments were performed. One experiment is a test designed to discriminate system-generated music from conventionally-composed music. Another experiment is to test whether the music style of the generated music is similar to that of the given music objects. Moreover, to evaluate the efficiency of the proposed approach, the other experiment was conducted to measure the elapsed time of music generation. At last, a case study is given to demonstrate an example generated by our system.

6.1 Turing-like test

It is difficult to evaluate the effectiveness of a computer-generated music composition system because the evaluation of effectiveness in works of art often comes down to subjective opinion. In 2001, M. Pearce addressed this problem and proposed a method to evaluate the computer music composition system [36]. This study has adopted this method to design the necessary experiments.

The proposed system can be considered successful if the subjects cannot distinguish the system-generated music from the human-composed music. There were 36 subjects, including four well-trained music experts. The prepared dataset consisted of 10 machine-generated music objects and 10 human-composed music objects. The latter comprised "Beyer 8", "Beyer 11", "Beyer 35", "Beyer 51", "Through All Night", "Beautiful May", "Listen to Angle Singing", "Melody", "Moonlight", and "Up to Roof." These music objects are all piano music containing melody and accompaniment. These 20 music objects were sorted randomly and displayed to the subjects. The subjects were asked to listen to each music object and to determine whether it is system-generated or human-composed music. The proportions of correctly discriminated music were calculated from the obtained result (Mean is the average of the accuracy). The significant test was performed with the one-sample t-test against 0.5 (the expected value if subjects discriminated randomly).

The results of the experimental test are shown in Table 1. The results show that it is difficult to discriminate the system-generated music objects from the human-composed ones. All the subjects (including the experts) displayed a higher degree of discrimination because some of them possess extensive musical backgrounds.

6.2 Effectiveness evaluation for styled composition

In the second experiment, an attempt was made to evaluate whether or not the music style of the system-generated music is similar to that of the given music. The proposed system was demonstrated for the subjects on the world-wide web: http://avatar.cs.nccu.edu.tw/~stevechiu/cms/experiment2. For each round of music generation, subjects were asked to give a score, from 0 to 3, to denote the degree to which they felt it was dissimilar or similar. Each subject repeated this process three times. A total of 31 subjects performed the test with a resulting mean score of 1.405 and a standard deviation of 0.779.

6.3 Efficiency evaluation of music generation

To evaluate the response time of the developed music composition system, the third experiment was conducted. The experiment was conducted on an IBM desktop computer with a 2.4 Ghz Intel(R) Pentium(R) quad-core processor with 4 gigabytes main memory running Linux 2.6 operating system.

Table 1 The results of the discrimination test

	Mean	SD	DF	t	P-value
All subjects	0.522	0.115	35	1.16	0.253
All subjects except experts	0.503	0.106	31	0.166	0.869

SD the standard deviation, *DF* the degree of freedom, *t* t statistic

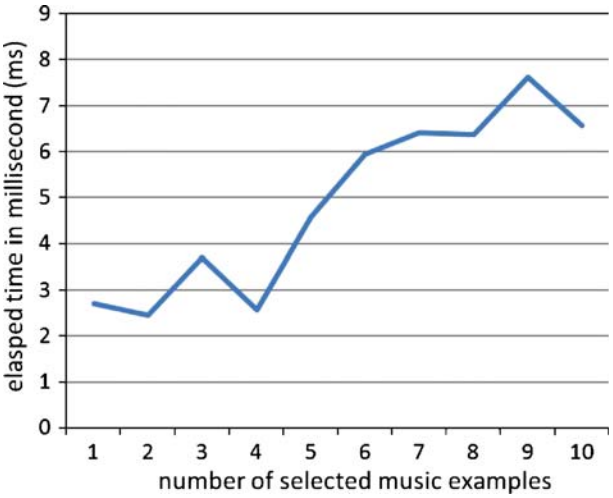

Fig. 15 Elapsed time of the online process of our system

There are 39 music objects collected from the Internet. The average number of notes per music in database was 145.2. The analysis component, especially the motif mining, takes most of the execution time in the whole process. While the analysis component was executed offline instead of online, the elapsed time for learning step and generating a new music object is shown in Fig. 15 as a function of the number of selected music examples. It can be seen that with the increasing number of selected music examples, the elapsed time of on line processing takes less than 10 ms.

6.4 Case study

The results of the proposed approach are shown by using an example. Using six music objects as input, "Beyer 55," "Grandfather's clock," "Little Bee," "Little Star," "My Family," and "Ode to Joy," the obtained result is an AABA form. The resulting phrase arrangements are 1-1-2 in Section A and 2-2 in Section B. At the melody style mining step,

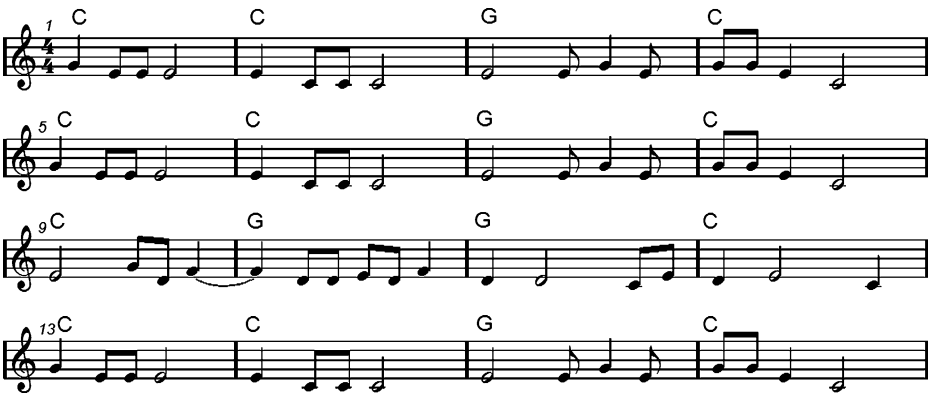

Fig. 16 An example of composed music using the proposed approach

the following patterns were discovered: {{C}, {G}, {C, G}}, {(G, C), (C, G)}, {<C, G>, <G, C>, <C, G, C>}. Furthermore, the chord generation component was found to generate the chord progressions <C, C, G, C> in Section A and <C, G, G, C> in Section B. The motif selection model chose a motif "sol-mi-mi" for generating the first phrase and the melody generation component develops this motif for the second phrase melody in Section A. Finally, the resulting musical composition is shown in Fig. 16.

7 Conclusions

In this paper, we have proposed a top-down approach for a music compositional system. Data mining techniques have been utilized to analyze and discover the common patterns or characteristics of music structure, melody style and motifs from the given musical pieces. The patterns discovered and the characteristics which constitute music structure, the melody style, and the motif selection model. The proposed system generates music based on these three models. The experimental results show that it is not easy to distinguish the system-generated music from the human-composed music. Future work includes of embedding other compositional elements such as rhythmic development, mode, and tone color into the composition process.

Acknowledgments We thank the anonymous referees for their valuable comments and suggestions.

References

1. Agrawal, R, Srikant R (1994) Fast algorithms for mining association rules. In: Proc of International Conference on Very Large Data Bases VLDB'94, 8
2. Agrawal R, Srikant R (1995) Mining sequential patterns. In: Proc. of IEEE International Conference on Data Engineering ICDE'95
3. Ame C, Domino M (1992) Cybernetic composer: an overview. In: Balaban M, Ebcioglu K, Laske O (ed) Understanding music with AI. AAAI
4. Bel B (1998) Migrating musical concepts: an overview of the bol processor. Comput Music J 22(2)
5. Biles JA (1994) GenJam: a genetic algorithm for generating jazz solos. In: Proc. of International Computer Music Conference, ICMC'94
6. Blackwell TM, Bentley P (2002) Improvised music with swarms. In: Proc. of the Congress on Evolutionary Computation CEC'02
7. Brown AR (2002) Opportunities for evolutionary music composition. In: Proc. of Australasian Computer Music Conference ACMA'02
8. Cambouropoulos E (2001) The local boundary detection model (LBDM) and its application in the study of expressive timing. In: Proc. of International Computer Music Conference ICMC'01
9. Chai W, Vercoe B (2001) Folk music classification using hidden markov models. In: Proc. of the International Conference on Artificial Intelligence ICAI'01
10. Chen CCJ, Miikkulainen R (2001) Creating melodies with evolving recurrent neural network. In: Proc. of International Joint Conference on Neural Networks IJCNN'01
11. Chen HC, Lin CH, Chen ALP (2004) Music segmentation by rhythmic features and melodic shapes. In: Proc. of IEEE International Conference on Multimedia and Expo ICME'04
12. Conklin D (2003) Music generation from statistical models. In: Proc. of the Symposium on Artificial Intelligence and Creativity in the Arts and Sciences AISB'03
13. Cope D (1991) Recombinant music using the computer to explore musical style. IEEE Computer 24(7)
14. Cope D (1992) Computer modeling of musical intelligence in EMI. Comput Music J 16(2)
15. Cope D (1996) Experiments in musical intelligence. A-R Editions
16. Cope D (2000) The algorithmic composer. A-R Editions

17. Correa DC, Levada ALM, Saito JH, Mari JF (2008) Neural network based systems for computer-aided musical composition: supervised x unsupervised learning. In: Proc. of ACM Symposium Applied Computing SAC'08
18. Dannenberg R, Thom B, Watson D (1997) A machine learning approach to musical style recognition. In: Proc. of International Computer Music Conference ICMC'97
19. Davie CT (1966) Musical structure and design. Dover
20. Dubnov S, Assayag G, Lartillot O, Gejerano G (2003) Using machine-learning methods for musical style modeling. IEEE Comput 36(10)
21. Espi D, Ponce de Leon PJ, Perez-Sancho C et al (2007) A cooperative approach to style-oriented music composition. In: Proc. of International Workshop on Artificial Intelligence and Music MUSIC-AI'07
22. Farbood M (2001) Analysis and synthesis of palestrina-style counterpoint using markov chains. In: Proc. of International Computer Music Conference ICMC'01
23. Guéret C, Monmarché N, Slimane M (2004) Ants can play music. In: Proc. of Ant Colony Optimization and Swarm Intelligence ANTS'04
24. Ho MC, Shan MK (2007) Theme-based music structural analysis. submitted to IEEE Multimedia
25. Hsu JL, Liu CC, Chen ALP (2001) Efficient repeating pattern finding in music database. IEEE Trans Multimed 3(3)
26. Jones NC, Pevzner PA (2004) An introduction to bioinformatics algorithms. The MIT
27. Karydis I, Nanopoulos A, Manolopoulos Y (2007) Finding maximum-length repeating patterns in music databases. Multimed Tools Appl 32(1)
28. Lartillot O, Dubnov S, Assayag G, Bejerano G (2002) Automatic modeling of musical style. In: Proc. of International Conference in Computer Music ICMC'02
29. Lo YL, Lee WL, Chang LH (2008) True suffix tree approach for discovering non-trivial repeating patterns in a music object. Multimed Tools Appl 37(2)
30. McAlpine K, Miranda ER, Hoggar S (1999) Composing music with algorithms: a case study system. Comput Music J 23(2)
31. Melucci M, Orio N (2002) A comparison of manual and automatic melody segmentation. In: Proc. of International Symposium on Music Information Retrieval ISMIR'02
32. Miranda ER (2001) Composing music with computers. Focal
33. Miura T, Tominaga K (2006) An approach to algorithmic music composition with an artificial chemistry. In: Proc. of the German Workshop on Artificial Life
34. Muscutt K (2007) Composing with algorithms: an interview with David Cope. Comput Music J 31(3)
35. Papadopoulos G, Wiggins G (1998) A genetic algorithm for generation of jazz melodies. In: Proc. of Software Technology and Engineering Practice STEP'98
36. Pearce M, Wiggins G (2001) Towards a framework for the evaluation of machine compositions. In: Proc. of Symposium on Artificial Intelligence and Creativity in the Arts and Sciences AISB'01
37. Ponce de Leon PJ, Inesta JM (2007) Pattern recognition approach for music style identification using shallow statistical descriptors. IEEE Trans Syst Man Cybern C Appl Rev 37(2)
38. Reagan P (1999) Computer music generation via decision tree learning. Master Thesis, Department of Computer Science, Carnegie Mellon University, USA
39. Shan MK, Kuo FF (2003) Music style mining and classification by melody. IEICE Trans Inf Syst E86-D (4)
40. Shan MK, Kuo FF, Chen MF (2002) Music style mining and classification by melody. In: Proc. of IEEE International Conference on Multimedia and Expo ICME'02
41. Sorensen A, Brown AR (2000) Introducing jMusic. In: Proc. of the Australasian Computer Music Conference
42. Stein L (1979) Structure & style: the study and analysis of musical forms. Summy-Birchard Music
43. Takasu A, Yanase T, Kanazawa T, Adachi J (1999) Music structure analysis and its application to theme phrase extraction. In: Proc. of European Conference on Research and Advanced Technology for Digital Libraries ECDL'99
44. Tominaga K ,Setomoto M (2008) An artificial-chemistry approach to generating polyphonic musical phrases. Lect Notes Comput Sci 4976
45. Uitdenbogerd AL, Zobel J (1999) Melodic matching techniques for large music databases, In: Proc. of ACM International Conference on Multimedia MM'99
46. Utgoff PE, Kirlin PB (2006) Detecting motives and recurring patterns in polyphonic music. In: Proc. of International Computer Music Conference ICMC'06
47. van Kranenburg P, Backer E (2004) Music style recognition—A quantitative approach. In: Proc. of Conference on Interdisciplinary Musicology CIM'04
48. Verbeurgt K, Dinolfo M, Fayer M (2004) Extracting patterns in music for composition via markov chains. In: Proc. of International Conference on Innovations in Applied Artificial Intelligence

49. Weyde T, Wissmann J, Neubarth K (2007) An experiment on the role of pitch intervals in melodic segmentation. In: Proc. of International Symposium on Music Information Retrieval ISMIR'07
50. Wiggins G, Papadopoulos G, Phon-Amnuaisuk S, Tuson A (1998) Evolutionary methods for musical composition. In: Proc. of Anticipation Music and Cognition CASYS'98
51. Witten IH, Frank E (2005) Data mining: Practical machine learning tools and techniques. Kaufmann, CA

Man-Kwan Shan received the B.S. degree in computer engineering and the M.S. degree in computer and information science both from National Chiao-Tung University, Taiwan, in 1986 and 1988, respectively. From 1988 to 1990, he served as a lecture in the Army Communications and Electronics School. Then, he worked as a lecture at the Computer Center of National Chiao-Tung University, where he supervised the Research and Development Division. He received the Ph.D. degree in Computer Science and Information Engineering from National Chiao-Tung University in 1998. Then he joined the Department of Computer Science at National Chengchi University as an assistant professor. He became an associated professor in 2003. His current research interests include data mining, multimedia systems and bioinformatics.

Shih-Chuan Chiu received the B.S. degree in Computer Science and Information Engineering from Tamkang University and the M.S. degree in Computer Science from National Chengchi University, in 2002 and 2004, respectively. He is currently working towards the Ph.D. degree in the Department of Computer Science at National Chiao-Tung University. His current research interests include data mining, computer music and mobile computing.