INTERACTIVE ARRANGEMENT OF CHORDS AND MELODIES BASED ON A TREE-STRUCTURED GENERATIVE MODEL

Hiroaki Tsushima **Eita Nakamura** **Katsutoshi Itoyama** **Kazuyoshi Yoshii**
Graduate School of Informatics, Kyoto University, Japan
{tsushima, enakamura}@sap.ist.i.kyoto-u.ac.jp, {itoyama, yoshii}@kuis.kyoto-u.ac.jp

ABSTRACT

We describe an interactive music composition system that assists a user in refining chords and melodies by generating chords for melodies (harmonization) and vice versa (melodization). Since these two tasks have been dealt with independently, it is difficult to jointly estimate chords and melodies that are optimal in both tasks. Another problem is developing an interactive GUI that enables a user to partially update chords and melodies by considering the latent tree structure of music. To solve these problems, we propose a hierarchical generative model consisting of (1) a probabilistic context-free grammar (PCFG) for chord symbols, (2) a metrical Markov model for chord boundaries, (3) a Markov model for melody pitches, and (4) a metrical Markov model for melody onsets. The harmonic functions (syntactic roles) and repetitive structure of chords are learned by the PCFG. Any variables specified by a user can be optimized or sampled in a principled manner according to a unified posterior distribution. For improved melodization, a long short-term memory (LSTM) network can also be used. The subjective experimental result showed the effectiveness of the proposed system.

1. INTRODUCTION

Music composition is a highly intelligent task that has been considered to be done only by musically trained people. To help musically untrained people create their own musical pieces, automatic music composition has actively been studied (*e.g.*, [4, 8, 19, 31]). While conventional studies have aimed at full automation of music composition, in the process of music composition, melodies (sequences of musical notes) and chord sequences are partially and incrementally refined by trial and error until the resulting musical piece has musically appropriate structure. Our aim is to develop an interactive arrangement system that can assist unskillful people to take such a process for reflecting their preference in creating melodies and chord sequences.

It is non-trivial to reflect user's preference to a musical piece in a consistent and unified framework of statistical

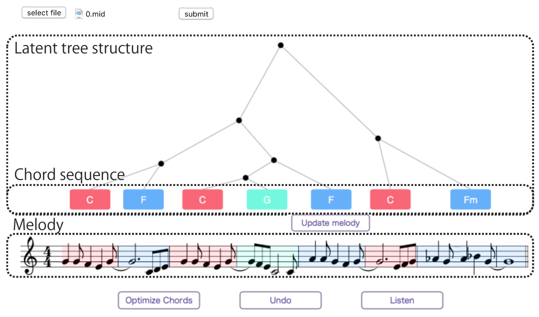


Figure 1: Our interactive music arrangement system based on a tree-structured generative model.

modeling. This problem is hard to solve especially when a black-box method (*e.g.*, neural end-to-end learning) is used for music generation. To incrementally refine a musical piece, one may iteratively use a harmonization method for generating a chord sequence from a melody [4, 19, 24, 28] and a melodization method for generating a melody from a chord sequence [3, 7, 8, 15, 22, 30, 31]. This approach, however, cannot enable a user to partially and incrementally refine melodies and chords in consideration of the optimality of the whole musical piece because each task has a unique evaluation criterion.

Since music is typically well-characterized by chords and melodies, it is important to be aware of complicated structures within and between chords and melodies. when composing a musical piece. To generate a musically appropriate sequence of chords, the harmonic functions of chords, which typically consist of three categories, *i.e.*, tonic (T), dominant (D), and subdominant (SD), should be considered because such functions represent syntactic roles in the same way as parts of speech in written texts. In addition, a sequence of harmonic functions of chords has a tree structure [21, 26]. For example, a chord sequence (C, Dm, G, Am, C, F, G, C) can be interpreted as (((T, SD), (D, T)), ((T, SD), (D, T))), where subtrees such as (T, SD), (D, T), and ((T, SD), (D, T)) appear repeatedly in a hierarchical manner. Therefore, it is desirable to consider such the hierarchical tree structure of chord sequences when we computationally help people to create a new music.

In this paper we propose an interactive music arrangement system that enables musically untrained users to create a melody and a chord sequence (Fig. 1). To partially and incrementally refine the piece, users can choose several types of operations that are often exploited by musically trained people. Specifically, the entire chord sequence and the corresponding tree structure can be refined jointly for a melody; the onset time of a specified chord can be refined; two adjacent chords forming a subtree can be merged into a single chord or a chord can be split into two chords; and melody notes in the region of a specified chord can be refined. All a user needs to do is to specify where to update the piece and it is not necessary to manually edit individual musical elements.

To optimize a chord sequence and a melody in a unified criterion, we propose a tree-structured hierarchical generative model that consists of (i) a probabilistic context-free grammar (PCFG) generating chord symbols [28], (ii) a metrical Markov model generating chord rhythms, and (iii) a Markov model generating melody pitches conditionally on the chord sequence, and (iv) a metrical Markov model generating melody rhythms (Fig. 2). The rule probabilities of the PCFG are learned from chord sequences, with the expectation that the syntactic roles of chords are captured by the non-terminal symbols [29]. The other models are also learned from chord and/or note sequences. To improve the melodization process, a long short-term memory (LSTM) network can be used instead of the Markov models (iii) and (iv) for capturing the long-term characteristic of a melody. Using the generative model trained in advance, we can estimate any "missing" variables, *i.e.*, an unpleasant part of chords or musical notes specified by the user, in a statistical manner.

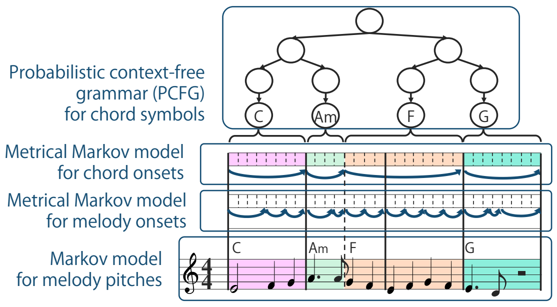
The major contribution of this study is the realization of a directability-aware music composition/arrangement system based on a unified probabilistic model. This system provides a user with an easy-to-use GUI that shows other possibilities for an unpleasant part of the piece and all operations on the GUI are implemented as posterior inference based on the probabilistic model. Our contribution lies in the marriage of AI and human creativity.

2. RELATED WORK

This section reviews related studies on automatic harmonization and melodization.

2.1 Automatic Harmonization

Many studies have been conducted for automatic harmonization for given melodies. Some studies aim to generate a sequence of chord symbols (as in this paper), and others aim to generate several (typically four) voices of musical notes. In the former type of research, Chuan and Chew [4] proposed a method consisting of three processes: selecting musical notes that might form chords from given melodies with a support vector machine (SVM), constructing triad chords from the selected notes, and generating chord progressions by using a rule-base method. Simon et al. [24] proposed a commercial system *MySong* based on hidden Markov models (HMMs) with Markovian chord transitions. Raczyński et al. [20] proposed similar Markov models in which chords are conditioned by melodies and time-varying keys. Tsushima et al. [28] proposed a harmonization method that considers the hierarchical repetitive structure of sequences of chord symbols obtained by


Probabilistic context-free grammar (PCFG) for chord symbols
Metrical Markov model for chord onsets
Metrical Markov model for melody onsets
Markov model for melody pitches
C
Am
F
G
Figure 2: A tree-structured hierarchical generative model for chord symbols and melodies.

PCFGs and pitch transitions conditioned by chord symbols with Markov models. De Prisco et al. [19] proposed a harmonization method for only a base line of the input with a distinctive network that models the dependencies among bass notes, the previous chord, and the current chord.

In the latter type of research, Ebcioğlu [6] proposed a rule-based method for generating four-part chorales in Bach's style. Several methods of using variants of genetic algorithms (GAs) based on music theories have also been proposed [17, 18, 27]. Allan and Williams [2] proposed an HMM-based method that represents chords as hidden states and musical notes as observed outputs. A hidden semi-Markov model (HSMM) [11] has been used for explicitly representing the durations of chords. Paiement et al. [16] proposed a hierarchical tree-structured model that describes chord movements from the viewpoint of hierarchical time scales by dividing the notations of chords. To generate highly convicting four-part chorales, a deep recurrent neural network has also been used for capturing the long-term characteristic of a melody and a harmony [12].

2.2 Automatic Melodization

There have been many studies on automatic melodization [3,8,15,22,30,31]. Fukayama et al. [8] developed a system named *Orpheus* that generates a melody for a given lyric in a way that the prosody of the lyric matches the dynamics of the melody. Roig et al. [22] proposed a method of generating a monophonic melody by using a probabilistic model of rhythm patterns and pitch contours.

Recent studies have applied deep learning techniques. In *Magenta* project [30], for example, recurrent neural networks (RNNs) are used for learning long-term dependency of music. Yang et al. [31] proposed a novel method for generating diverse monophonic melodies by combining a generative adversarial network (GAN) with a convolutional neural network (CNN). To generate diverse melodies, Mogren [15] proposed adversarial training of an RNN that works on continuous sequential data. The method based on a restricted Boltzmann machine (RBM) conditioned on RNNs that models temporal dependency has been proposed to generate polyphonic music [3]. In addition, Eck et al. [7] have proposed an LSTM-based method for generating both melodies and chords by capturing the characteristic of note-by-note transitions and the mutual dependency between musical notes and chord symbols.

3. USER INTERFACE

The proposed system, which is implemented as a web service based on HTML5, enables a user to incrementally refine a chord sequence and a melody on a GUI (Fig. 1). To use a system, a user is asked to upload a melody of eight bars. The system then estimates a chord sequence that harmonizes with the melody. The chord onsets are located at the bar lines. Supported arrangement operations are:

- **Updating the chord symbols**: The chord symbols and the latent tree structure behind the chord symbols are jointly optimized for the current melody.
- **Updating a chord onset**: One of the chord onsets (boundaries) specified by a user is optimized.
- **Splitting a chord**: One of the chords specified by a user is split into two adjacent chords.
- **Merging chords**: Two adjacent chords that form a subtree are merged into a single chord.
- **Updating the melody**: Melody notes in the region of a chord specified by a user are updated while keeping consistency with neighboring measures.

4. PROBABILISTIC MODELING

This section explains a unified probabilistic model that represents the hierarchical generative process of a chord sequence and a melody. The proposed model consists of four sub-models, which are trained independently.

4.1 Mathematical Notation

We assume that chord and melody onsets are on the 16th-note-level grid. Let L be the number of measures of a musical piece ($L = 8$ in this paper) and $T = 16L$ be the total number of time units. A sequence of chord symbols and that of chord onsets are denoted by $\boldsymbol{z} = \{z_n\}_{n=1}^N$ and $\boldsymbol{\phi} = \{\phi_n\}_{n=1}^N$, respectively, where N is the number of chords and ϕ_n takes an integer in $[0, T)$. Similarly, a sequence of melody pitches and that of melody onsets in the region of chord z_n is denoted by $\boldsymbol{p}_n = \{p_{n,i}\}_{i=1}^{I_n}$ and $\boldsymbol{\psi}_n = \{\psi_{n,i}\}_{i=1}^{I_n}$, respectively, where I_n is the number of musical notes in that time span, $p_{n,i}$ is a MIDI note number from 32 to 93, and $\psi_{n,i}$ takes an integer in $[\phi_n, \phi_{n+1})$. The whole melody is denoted by $\boldsymbol{p} = \{\boldsymbol{p}_n\}_{n=1}^N$ and $\boldsymbol{\psi} = \{\boldsymbol{\psi}_n\}_{n=1}^N$, where $I = \sum_{n=1}^N I_n$ is the number of melody notes.

Let $\boldsymbol{t}$ be a latent tree that derives $\boldsymbol{z}$ according to a PCFG and $t_{m:n}$ be an *inside* part (subtree) of $\boldsymbol{t}$ that derives $z_{m:n}$. Thus $\boldsymbol{t} = t_{1:N}$. We often use $t_{m:n}$ to indicate the root node of the subtree for simplicity. Let $t_{\neg m:n}$ be an *outside* part of $\boldsymbol{t}$ that derives $z_{1:m-1}$, $t_{m:n}$, and $z_{n+1:N}$.

4.2 Model Formulation

We formulate a unified probabilistic model that represents the generative process of a latent tree $\boldsymbol{t}$, chord symbols $\boldsymbol{z}$, chord onsets $\boldsymbol{\phi}$, melody pitches $\boldsymbol{p}$, and melody onsets $\boldsymbol{\psi}$.

4.2.1 Probabilistic Context-Free Grammar for $\boldsymbol{t}$ *and* $\boldsymbol{z}$

A derivation tree $\boldsymbol{t}$ and chord symbols $\boldsymbol{z}$ are generated in this order according to a PCFG $G = (V, \Sigma, R, S)$, defined

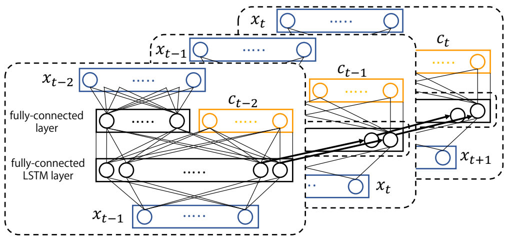


Figure 3: Configuration of the LSTM network

by a set of non-terminal symbols V that are expected to represent the hierarchical structure and syntactic roles of chords, a set of terminal symbols (chord symbols) Σ, a set of rule probabilities R, and a start symbol S (a non-terminal symbol located on the root of a syntax tree). There are three types of rule probabilities. $\theta_{A \to BC}$ is the probability that a non-terminal symbol $A \in V$ branches to non-terminal symbols $B \in V$ and $C \in V$. $\eta_{A \to \alpha}$ is the probability that $A \in V$ emits terminal symbol $\alpha \in \Sigma$. A non-terminal symbol $A \in V$ emits a terminal symbol with a probability of $0 < \lambda_A < 1$ and otherwise it branches. These probabilities are normalized as follows:

$$\sum_{B,C \in V} \theta_{A \to BC} = 1, \quad \sum_{\alpha \in \Sigma} \eta_{A \to \alpha} = 1. \tag{1}$$

We let $\boldsymbol{\theta}_A = \{\theta_{A \to BC}\}_{B,C \in V}$ and $\boldsymbol{\eta}_A = \{\eta_{A \to \alpha}\}_{\alpha \in \Sigma}$.

4.2.2 Metrical Markov Models for $\boldsymbol{\phi}$ *and* $\boldsymbol{\psi}$

The metrical Markov model for chord onsets $\boldsymbol{\phi}$ on the regular 16th-note-level grid is defined by

$$p(\phi_n | \phi_{n-1}) = \pi_{\phi_{n-1} \bmod 16, \phi_n - \phi_{n-1}}, \tag{2}$$

where $\pi_{a,b}$ indicates the probability that a chord starting at the a-th position in a measure ($0 \le a < 16$) continues for the duration of b time units ($0 < b \le T$).

A similar model for melody onsets $\boldsymbol{\psi}$ is defined by

$$p(\psi_{n,1} | \psi_{n-1,I_{n-1}}) = \rho_{\psi_{n-1,I_{n-1}} \bmod 16, \psi_{n,1} - \psi_{n-1,I_{n-1}}},$$
$$p(\psi_{n,i} | \psi_{n,i-1}) = \rho_{\psi_{n,i-1} \bmod 16, \psi_{n,i} - \psi_{n,i-1}} \ (1 < i), \tag{3}$$

where $\rho_{a,b}$ indicates the probability that a musical note starts at the a-th position in a measure ($0 \le a < 16$) and continues for the duration of b time units ($0 < b \le T$).

4.2.3 Markov Model for $\boldsymbol{p}$ *Conditioned on* $\boldsymbol{z}$

The Markov model for melody pitches $\boldsymbol{p}$ conditioned by a chord sequence given by $\boldsymbol{z}$ is defined by

$$p(p_{n,1} | p_{n-1,I_{n-1}}, z_n) = \tau^{z_n}_{p_{n-1,I_{n-1}}, p_{n,1}}, \tag{4}$$

$$p(p_{n,i} | p_{n,i-1}, z_n) = \tau^{z_n}_{p_{n,i-1}, p_{n,i}} \ (2 \le i \le I_n), \tag{5}$$

where $\tau^c_{a,b}$ is the transition probability from pitch a to pitch b under chord symbol c.

4.2.4 Bayesian Integration of Four Sub-models

Letting $\boldsymbol{\Omega} = \{\boldsymbol{t}, \boldsymbol{z}, \boldsymbol{\phi}, \boldsymbol{p}, \boldsymbol{\psi}\}$ be a set of the external random variables and $\boldsymbol{\Theta} = \{\boldsymbol{\theta}, \boldsymbol{\eta}, \boldsymbol{\lambda}, \boldsymbol{\pi}, \boldsymbol{\rho}, \boldsymbol{\tau}\}$ be a set of the model parameters, the unified model is given by

$$p(\boldsymbol{\Omega}, \boldsymbol{\Theta}) = p(\boldsymbol{t}, \boldsymbol{z} | \boldsymbol{\theta}, \boldsymbol{\eta}, \boldsymbol{\lambda}) p(\boldsymbol{\phi} | \boldsymbol{\pi}) p(\boldsymbol{\psi} | \boldsymbol{\tau}) p(\boldsymbol{p} | \boldsymbol{z}) p(\boldsymbol{\Theta}), \tag{6}$$

where $p(\boldsymbol{\Theta}) = p(\boldsymbol{\theta}) p(\boldsymbol{\eta}) p(\boldsymbol{\lambda}) p(\boldsymbol{\pi}) p(\boldsymbol{\rho}) p(\boldsymbol{\tau})$ is a prior distribution over $\boldsymbol{\Theta}$. To make Bayesian inference tractable, we use conjugate Dirichlet and beta priors as follows:

$$\boldsymbol{\theta}_A \sim \mathrm{Dir}(\boldsymbol{\xi}_A),\ \boldsymbol{\eta}_A \sim \mathrm{Dir}(\boldsymbol{\zeta}_A),\ \lambda_A \sim \mathrm{Beta}(\boldsymbol{\iota}_A), \quad (7)$$

$$\boldsymbol{\pi}_a \sim \mathrm{Dir}(\boldsymbol{\beta}_a),\ \boldsymbol{\rho}_a \sim \mathrm{Dir}(\boldsymbol{\gamma}_a),\ \boldsymbol{\tau}_a^c \sim \mathrm{Dir}(\boldsymbol{\delta}_a^c), \quad (8)$$

where $\boldsymbol{\xi}_A$, $\boldsymbol{\zeta}_A$, $\boldsymbol{\iota}_A$, $\boldsymbol{\beta}_a$, $\boldsymbol{\gamma}_a$, and $\boldsymbol{\delta}_a^c$ are hyperparameters.

4.2.5 LSTM Network for $\boldsymbol{x}$ Conditioned on $\boldsymbol{c}$

In melody arrangement, we can also use an LSTM model that can learn complicated long-term dynamics of melodies. Let $\boldsymbol{x} = \{x_t\}_{t=1}^T$ be another representation of the entire melody, where x_t takes a MIDI note number at the t-th position ($0 \le t < T$) if the note onset is at that position and otherwise takes 0. Let $\boldsymbol{c} = \{c_t\}_{t=1}^T$ be another representation of the entire chord sequence given by $\boldsymbol{z}$ and $\boldsymbol{\phi}$, where c_t indicates a chord symbol at the t-th position. Given a sequence of musical notes $x_{1:t} = \{x_i\}_{i=1}^t$ and that of chord symbols $c_{1:t} = \{c_i\}_{i=1}^t$, the LSTM model determines the probability of the next musical note given by $p(x_{t+1}|x_{1:t}, c_{1:t})$ (Fig. 3).

4.3 Model Training

Our goal is to obtain the maximum a posteriori (MAP) estimates of the model parameters $\Theta = \{\boldsymbol{\theta}, \boldsymbol{\eta}, \boldsymbol{\lambda}, \boldsymbol{\pi}, \boldsymbol{\rho}, \boldsymbol{\tau}\}$. To estimate the parameters $\boldsymbol{\theta}$, $\boldsymbol{\eta}$, and $\boldsymbol{\lambda}$ of the PCFG from a chord sequence $\boldsymbol{z}$ (multiple sequences are used in practice) in an unsupervised manner, we use an inside-filtering-outside-sampling algorithm [13,28] for generating samples from the true posterior distribution $p(\boldsymbol{\theta}, \boldsymbol{\eta}, \boldsymbol{\lambda}, \boldsymbol{t}|\boldsymbol{z})$. More specifically, the latent tree $\boldsymbol{t}$ and the parameters $\boldsymbol{\theta}$, $\boldsymbol{\eta}$, and $\boldsymbol{\lambda}$ are alternately sampled from the conditional posterior distributions $p(\boldsymbol{t}|\boldsymbol{\theta}, \boldsymbol{\eta}, \boldsymbol{\lambda}, \boldsymbol{z})$ and $p(\boldsymbol{\theta}, \boldsymbol{\eta}, \boldsymbol{\lambda}|\boldsymbol{t}, \boldsymbol{z})$, respectively.

The parameters $\boldsymbol{\pi}$, $\boldsymbol{\tau}$ and $\boldsymbol{\rho}$ of the Markov models are learned independently. Given a sequence of chord onsets $\boldsymbol{\phi}$ and a sequence of melody onsets $\boldsymbol{\psi}$, the posterior distribution of $\boldsymbol{\pi}$ and that of $\boldsymbol{\rho}$ can be calculated, respectively, because of the conjugacy between the Dirichlet and categorical distributions. Similarly, given a sequence of melody pitches $\boldsymbol{p}$ associated with a chord sequence specified by $\boldsymbol{z}$ and $\boldsymbol{\phi}$, the posterior distribution of $\boldsymbol{\tau}$ can be calculated. The LSTM network is also trained from the same data.

5. CHORD AND MELODY ARRANGEMENT

This section explains how to leverage the unified model described in Section 4 for implementing the five operations described in Section 3. Let $\boldsymbol{\Omega} = \{\boldsymbol{t}, \boldsymbol{z}, \boldsymbol{\phi}, \boldsymbol{p}, \boldsymbol{\psi}\}$ be a set of random variables. To estimate a "missing" part $\boldsymbol{\chi} \subset \boldsymbol{\Omega}$, we take a principled statistical approach based on the conditional posterior distribution $p(\boldsymbol{\chi}|\boldsymbol{\Omega}_{\neg\boldsymbol{\chi}}, \Theta)$, where $A_{\neg B}$ indicates a subset of A obtained by removing the elements of B from A. Note that full automatic music composition can be achieved by sampling $\boldsymbol{\Omega}$ from $p(\boldsymbol{\Omega}|\Theta)$.

5.1 Updating the Chord Symbols

When the melody pitches $\boldsymbol{p}$ are fixed, the chord symbols $\boldsymbol{z}$ and the latent tree $\boldsymbol{t}$ can be optimized by maximizing the conditional posterior distribution $p(\boldsymbol{t}, \boldsymbol{z}|\boldsymbol{p}, \Theta)$. Since both $\boldsymbol{t}$ and $\boldsymbol{z}$ are latent variables in this operation, we extend the Viterbi algorithm to infer $\boldsymbol{t}$ and $\boldsymbol{z}$ from $\boldsymbol{p}$. First, the inside

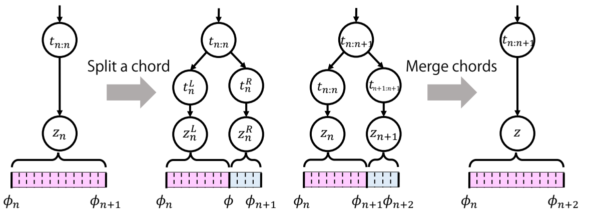


Figure 4: Split and merge operations.

probabilities are recursively calculated from the layer of terminal symbols $\boldsymbol{z}$ to the start symbol S according to

$$p_{n,n}^A = \lambda_A \max_{z \in \Sigma} \eta_{A \to z}\, p(\boldsymbol{p}_n|z), \quad (9)$$

$$p_{n,n+k}^A = (1-\lambda_A) \max_{\substack{B,C \in V \\ 1 \le l \le k}} \theta_{A \to BC}\, p_{n,n+l-1}^B p_{n+l,n+k}^C, \quad (10)$$

where $p(\boldsymbol{p}_n|z_n)$ is the probability that a pitch subsequence $\boldsymbol{p}_n$ is generated conditionally on chord z_n:

$$p(\boldsymbol{p}_n|z_n) = \prod_{i=1}^{I_n} p(p_{n,i}|p_{n,i-1}, z_n), \quad (11)$$

where $p_{n,0} = p_{n-1,I_{n-1}}$. The most likely $\boldsymbol{t}$ and $\boldsymbol{z}$ are obtained by recursively back-tracking the most likely paths from the start symbol S.

5.2 Updating a Chord Onset

When the melody pitches $\boldsymbol{p}$ and the melody onsets $\boldsymbol{\psi}$ are given and the chord symbols $\boldsymbol{z}$ are fixed, a chord onset ϕ_n can be optimized by maximizing the conditional posterior distribution given by

$$p(\phi_n|\boldsymbol{z}, \boldsymbol{\phi}_{\neg n}, \boldsymbol{p}, \boldsymbol{\psi}, \Theta)$$
$$\propto p(\boldsymbol{p}_{n-1}|z_{n-1})p(\boldsymbol{p}_n|z_n)p(\phi_n|\phi_{n-1})p(\phi_{n+1}|\phi_n), \quad (12)$$

where ϕ_n is restricted such that $\psi_{n-1,1} \le \phi \le \psi_{n,I_n}$.

5.3 Splitting a Chord and Merging Chords

The chord symbols $\boldsymbol{z}$ and the chord onsets $\boldsymbol{\phi}$ can be locally refined by splitting a chord into adjacent chords or merging adjacent chords into another chord (Fig. 4). A subtree of $\boldsymbol{t}$ is updated accordingly. The split operation can be applied to any chord z_n while the merge operation is restricted to adjacent chords $z_{n:n+1}$ forming a subtree $t_{n:n+1}$.

A chord z_n associated with a non-terminal symbol $t_{n:n}$ is split at a 16th-note-level position ϕ into two new chords z_n^L and z_n^R associated with two new symbols t_n^L and t_n^R by maximizing the conditional posterior distribution given by $p(t_n^L, t_n^R, z_n^L, z_n^R, \phi|t_{\neg n:n}, \boldsymbol{z}_{\neg n}, \boldsymbol{\phi}, \boldsymbol{p}, \boldsymbol{\psi}, \Theta)$. This operation makes a new subtree that has $t_{n:n}$ as its root node, derives t_n^L and t_n^R, and generates z_n^L and z_n^R. To do this, we use the extended Viterbi algorithm for estimating the most likely subtree from $\boldsymbol{p}_n$. First, the inside probabilities are recursively calculated from the layer of terminal symbols z_n^L and z_n^R to the root node $t_{n:n}$ according to

$$\alpha_\phi^A = \lambda_A \max_{z \in \Sigma} \eta_{A \to z}\, p(\boldsymbol{p}_n^L|z, \phi), \quad (13)$$

$$\beta_\phi^A = \lambda_A \max_{z \in \Sigma} \eta_{A \to z}\, p(\boldsymbol{p}_n^R|z, \phi), \quad (14)$$

$$p_\phi^{t_{n:n}} = \max_{B,C \in V} \theta_{t_{n:n} \to BC}\, \alpha_\phi^B \beta_\phi^C p(\phi|\phi_n)p(\phi_{n+1}|\phi), \quad (15)$$

where $\boldsymbol{p}_n^L$ and $\boldsymbol{p}_n^R$ are the subsequences of pitches obtained by splitting $\boldsymbol{p}_n$ with a boundary ϕ. The most likely z_n^L, z_n^R, t_n^L, t_n^R, and ϕ are obtained by recursively back-tracking the most likely paths from $t_{n:n}$.

Two adjacent chords z_n and z_{n+1} associated with non-terminal symbols $t_{n:n}$ and $t_{n+1:n+1}$ are merged into a single chord z associated with a non-terminal symbol $t_{n:n+1}$ by maximizing the conditional posterior distribution given by $p(z|t_{\neg n:n+1}, \boldsymbol{z}_{\neg n:n+1}, \boldsymbol{\phi}_{\neg n+1}, \boldsymbol{p}, \boldsymbol{\psi}, \boldsymbol{\Theta})$. The most likely z is obtained as follows:

$$z = \underset{z' \in \Sigma}{\arg\max}\, \eta_{t_{n:n+1} \rightarrow z'}\, p(\boldsymbol{p}_n|z')p(\boldsymbol{p}_{n+1}|z'). \quad (16)$$

5.4 Updating the Melody

When a chord symbol z_n, the last pitch $p_{n-1,I_{n-1}}$ in the region of the previous chord z_{n-1}, and the first pitch $p_{n+1,1}$ in the region of the next chord z_{n+1} are given, a sequence of musical notes in the region of z_n (between ϕ_n and ϕ_{n+1}) is obtained by maximizing the conditional posterior distribution $p(\boldsymbol{p}_n|z_n, p_{n-1,I_{n-1}}, p_{n+1,1}, \boldsymbol{\Theta})$. To do this, we propose an efficient algorithm based on dynamic programming. Let α_{y_t,d_t} be the marginal likelihood that a note at the pitch y_t is located on the score time t and the duration of the previous note is d_t on a chord z_n:

$$\alpha_{y_t,d_t} = p(y_t, d_t|z_n) \quad (17)$$

This probability can be calculated recursively in the score time $t \in \{\phi_n, ..., \phi_{n+1}, \psi_{n+1,1}\}$.

$$\alpha_{y_t,d_t} = \rho_{t-d_t,t} \sum_{y_{t-d_t},\, d_{t-d_t}} \alpha_{y_{t-d_t},d_{t-d_t}} \tau^{z_n}_{y_{t-d_t},y_t}$$

In each score time t, d_t can take values in $\{1, ..., t, t - \psi_{n-1,I_{n-1}}\}$. By using this probability, we can recursively sample $\boldsymbol{p}_n$ from the beat score time $\psi_{n+1,1}$ to $\psi_{n-1,I_{n-1}}$.

Another improved way of partially updating the melody is to use the LSTM model. Suppose that we aim to update $x_{i:j}$ in the whole melody $\boldsymbol{x}$. Given a chord sequence $\boldsymbol{c}$ and melody segments $x_{1:i-1}$ and $x_{j+1:T}$, the missing part $x_{i:j}$ can be sampled from the conditional posterior distribution $p(x_{i:j}|\boldsymbol{c}, x_{1:i-1}, x_{j+1:T}) \propto p(\boldsymbol{x}|\boldsymbol{c})$. First, the pitches $x_{1:i-1}$ and chords $c_{1:i-1}$ are fed to the network to update the hidden states. The missing part $x_{i:j}$ is then sampled sequentially according to the probability $p(x_{t+1}|x_{1:t}, c_{1:t})$ learned by the LSTM. This enables us to evaluate $p(\boldsymbol{x}|\boldsymbol{c})$. Among a sufficient number of generated samples of $x_{1:i-1}$, a sample with the highest $p(\boldsymbol{x}|\boldsymbol{c})$ is selected.

6. EVALUATION

This section reports objective and subjective evaluations on the user interface and the music arrangement method.

6.1 Experimental Conditions

To train the PCFG, we used 705 chord sequences of musical sections (e.g., verse, bridge, and chorus) from 468 pieces of popular music included in the SALAMI dataset [25]. Only chord sequences with a length between 8 and 32 measures were chosen. The vocabulary of chord symbols was limited to the combinations of the 12 root notes {C, C#, ..., B} and the 2 chord types {major, minor}. The number of kinds of non-terminal symbols of the PCFG was set to 12. The values of the hyperparameter $\boldsymbol{\iota}_A$ were all set to 1.0 and those of the other parameters were all set to 0.1. To train the three Markov models, we used 9902 pairs of melodies and the corresponding chord sequences from 194 pieces of popular music included in Rock Corpus [5]. To train the LSTM, we used 9265 melodies associated with chord sequences from pieces of popular music included in Rock Corpus and Nottingham Database [1]. Note that all of the data used in our experiments were transposed to the C major or C minor key. The number of the hidden units was 50 and the softmax-cross-entropy was used as a loss function. The parameters of the LSTM were optimized by using Adam [14]. The number of samples generated by the LSTM (described in Section 5.4) was 50.

6.2 Objective Evaluation of Melody Arrangement

We evaluated the function of updating a melody in terms of the note density of the generated musical notes via 10-fold cross validation on the Rock Corpus and Nottingham Database. For the region of each chord z_n, a sequence of melody $\boldsymbol{p}_n$ is arranged by using the two methods based on the Markov model and the LSTM described in Section 5.4. We measured the mean squared error (MSE) between the note density per measure of the generated musical notes and the mean value of the density of other regions given by

$$\mathrm{MSE} = \frac{1}{N-1} \sum_{n=1}^{N-1} \left\{ \frac{16 I_n^*}{\phi_{n+1} - \phi_n} - \frac{\sum_{m \neq n} 16 I_m}{\sum_{m \neq n} (\phi_{m+1} - \phi_m)} \right\}^2,$$

where I_n^* and I_n were the number of generated musical notes and that of the original musical notes, respectively. The average MSE was calculated over all melodies. The average MSE obtained by the LSTM model was 5.52 while that obtained by the Markov model was 6.42. This indicates that the LSTM-based method is a little more effective for updating a partial melody in consideration of the note density of the whole melody because it can capture the long-term dependency.

6.3 Subjective Evaluation of the Proposed System

We conducted the subjective evaluation of the system[1] in terms of usability and effectiveness in interactive chord and melody arrangement. Five melodies of 8 measures were extracted from the RWC music database [9, 10]. We asked 11 subjects to test our system. Four subjects who had the experience of playing musical instruments for more than five years were regarded as people with musical backgrounds. Each subject was asked to interactively make a musical piece by using each of the five melodies as an initial seed and then grade the system on a 5-point Likert scale (from "strongly agree (1)" to "strongly disagree (5)") in terms of the following 15 criteria:

- The chord sequences obtained were suitable for the melodies (I).
- The chord sequences obtained by the split or merge operation were musically natural (II, III).

[1] The interface used in this experiment is available online: http://sap.ist.i.kyoto-u.ac.jp/members/tsushima/ismir2018/

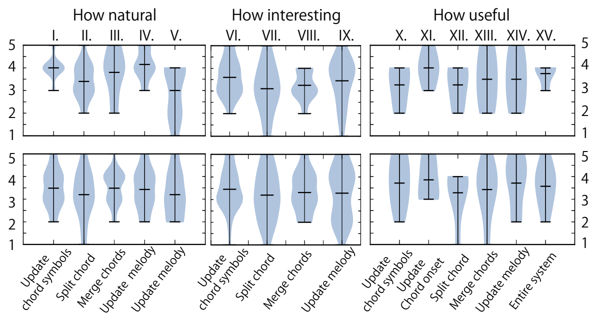


Figure 5: Results for people with musical backgrounds (top) and those for people without musical backgrounds (bottom). The middle bars indicate the mean value.

- The melodies obtained were suitable for the chord sequences (Ⅳ).
- The melodies obtained were musically natural (V).
- The musical pieces obtained by updating chord symbols, splitting a chord, merging chords, or updating a melody were interesting (Ⅵ, Ⅶ, Ⅷ, Ⅸ).
- The function of updating chord symbols, splitting a chord, merging chords, or updating a melody was useful (X, Ⅺ, Ⅻ, XIII, XIV).
- The user interface has the capability of helping users make musical pieces (XV).

We also asked the subjects to tell us how each of them felt about the system.

The results of this user study is shown in Fig 5. In terms of the naturalness and the interestingness, the two operations, updating chord symbols and updating melodies, obtained the slightly high mean ratings of 3.67 in criterion (I), 3.69 in criterion (Ⅳ), and 3.51 in criterion (Ⅵ). As seen in the score for the criterion (V), the subjects with musical backgrounds, compared with the others, tended to feel that the updated melodies were less musically natural.As seen in the score for the criterion (Ⅸ), the subjects with musical backgrounds tended to feel that the updated melodies were more interesting. In terms of the usefulness of each operation, each operation obtained the reasonably high mean ratings (from 3.27 to 3.91).

We obtained the following opinions on the usability of our system:

- It was interesting that even a user without any experiences in music composition can edit a musical piece by iterating several operations.
- An operation that updates one chord symbol is necessary for more freely editing a chord sequence.

We also obtained the following opinions on the problems of some operations:

- The chord sequences obtained were almost always appropriate for all samples of melodies but the system tended to generate only basic chords (e.g., C major).
- The updated melodies were often unnatural when an original melody has some repeated sections.

The reason for the former problem may be that the chord symbols are updating by using the Viterbi algorithm. The reason for the latter problem is probably that the LSTM cannot capture the global repetitive structure of a melody.

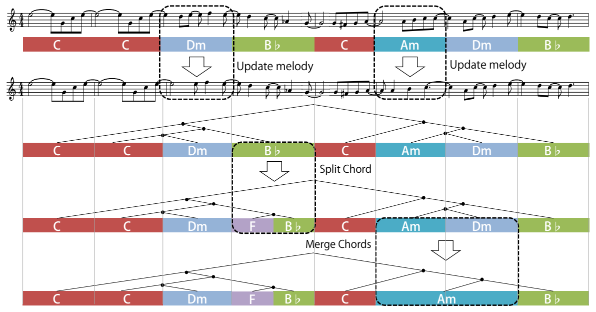


Figure 6: Example operation for interactive generation of chord sequences and melodies.

6.4 Example of Chord and Melody Arrangement

Fig. 6 shows how the proposed method generates chord sequences and melodies. The score (melodies and chords) at the top shows an initial state in which the chord symbols were optimized for the melody in the input file (the chord onsets were located at the bar lines). The second score shows the state in which the two regions of the melody under the 3rd and 6th chords were updated in order. The third chord sequence shows the state in which the 4th chord, B♭ major, was split into F major and B♭ major. The fourth chord sequence shows the state in which the 7th chord, A minor, and the 8th chord, D minor, were merged into A minor. This indicates that the proposed method can successfully help a user partially update a melody while keeping the consistency of the whole melody and that it can generate a chord sequence by considering the latent tree structure behind the chord sequence.

7. CONCLUSION

This paper presented an interactive music arrangement system that enables a user to incrementally refine a chord sequence and a melody. The experimental results showed that the proposed system has a great potential to help a user create his or her original musical pieces.

There would be much room for improving our method. To improve the diversity of generated chord symbols, the use of some sampling or beam-search method would be effective. To improve the naturalness of generated melodies, the use of a bidirectional LSTM [23] would be effective for considering the repetitive structures of melodies.

For more specific studies on the effectiveness of our system, we plan to measure how well test users can incrementally refine a musical piece compared with the conventional methods, by counting the number of necessary operations to make musical pieces meet their satisfaction. We also plan to conduct large-scale user studies of the system on the Web. Collecting time-series data of users' operations and created pieces, it would be possible to infer their musical preference and improve the model by reinforcement learning. Using the same data, it would be possible to reveal the process of music creation by humans in terms of edit operations and optimization strategies.

Acknowledgements: This study was partially supported by JST ACCEL No. JPMJAC1602, JSPS KAKENHI No. 26700020 and No. 16H01744, and Grant-in-Aid for JSPS Research Fellow No. 16J05486.

8. REFERENCES

[1] ABC version of the Nottingham music database. http://abc.sourceforge.net/NMD/.

[2] M. Allan and C. Williams. Harmonising chorales by probabilistic inference. In *NIPS*, pages 25–32, 2005.

[3] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. In *ICML*, 2012.

[4] C. H. Chuan and E. Chew. A hybrid system for automatic generation of style-specific accompaniment. In *IJWCC*, pages 57–64, 2007.

[5] T. D. Clercq and D. Temperley. A corpus analysis of rock harmony. *Popular Music*, 30(01):47–70, 2011.

[6] K. Ebcioğlu. An expert system for harmonizing four-part chorales. *Computer Music Journal*, 12(3):43–51, 1988.

[7] D. Eck and J. Schmidhuber. A first look at music composition using LSTM recurrent neural networks. *IDSIA*, 103(07-02), 2002.

[8] S. Fukayama et al. Orpheus: Automatic composition system considering prosody of Japanese lyrics. In *ICMC*, pages 309–310. Springer, 2009.

[9] M. Goto. AIST annotation for the RWC music database. In *ISMIR*, pages 359–360, 2006.

[10] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka. RWC music database: Popular, classical and jazz music databases. In *ISMIR*, pages 287–288, 2002.

[11] R. Groves. Automatic harmonization using a hidden semi-Markov model. In *AIIDE*, pages 48–54, 2013.

[12] G. Hadjeres and F. Pachet. DeepBach: A steerable model for Bach chorales generation. In *ICML*, pages 1362–1371, 2017.

[13] M. Johnson, T. L. Griffiths, and S. Goldwater. Bayesian inference for PCFGs via Markov chain Monte Carlo. In *NAACL-HLT*, pages 139–146, 2007.

[14] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICMR*, pages 1–15, 2014.

[15] O. Mogren. C-RNN-GAN: Continuous recurrent neural networks with adversarial training. In *Constructive Machine Learning Workshop (NIPS 2016)*, 2016.

[16] J. F. Paiement, D. Eck, and S. Bengio. Probabilistic melodic harmonization. In *CSCSI*, pages 218–229, 2006.

[17] G. Papadopoulos and G. Wiggins. AI methods for algorithmic composition: A survey, a critical view and future prospects. In *AISB Symposium on Musical Creativity*, pages 110–117, 1999.

[18] R. D. Prisco and R. Zaccagnino. An evolutionary music composer algorithm for bass harmonization. In *Applications of Evolutionary Computing*, pages 567–572. Springer, 2009.

[19] R. De Prisco, A. Eletto, A. Torre, and R. Zaccagnino. A neural network for bass functional harmonization. In *European Conference on the Applications of Evolutionary Computation*, pages 351–360. Springer, 2010.

[20] S. A. Raczyński, S. Fukayama, and E. Vincent. Melody harmonization with interpolated probabilistic models. *Journal of New Music Research*, 42(3):223–235, 2013.

[21] M. Rohrmeier. Mathematical and computational approaches to music theory, analysis, composition and performance. *Journal of Mathematics and Music*, 5(1):35–53, 2011.

[22] C. Roig, L. J. Tardón, T. Barbancho, and A. M. Barbancho. Automatic melody composition based on a probabilistic model of music style and harmonic rules. *Knowledge-Based Systems*, 71:419–434, 2014.

[23] M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[24] I. Simon, D. Morris, and S. Basu. Mysong: automatic accompaniment generation for vocal melodies. In *SIGCHI Conference on Human Factors in Computing Systems*, pages 725–734. ACM, 2008.

[25] J. B. L. Smith, J. A. Burgoyne, I. Fujinaga, D. D. Roure, and J. S. Downie. Design and creation of a large-scale database of structural annotations. In *ISMIR*, pages 555–560, 2011.

[26] M. J. Steedman. A generative grammar for jazz chord sequence. *Music Perception*, 2(1):52–77, 1984.

[27] M. Towsey, A. Brown, S. Wright, and J. Diederich. Towards melodic extension using genetic algorithms. *Educational Technology & Society*, 4(2):54–65, 2001.

[28] H. Tsushima, E. Nakamura, K. Itoyama, and K. Yoshii. Function- and rhythm-aware melody harmonization based on tree-structured parsing and split-merge sampling of chord sequences. In *ISMIR*, pages 502–508, 2017.

[29] H. Tsushima, E. Nakamura, K. Itoyama, and K. Yoshii. Generative statistical models with self-emergent grammar of chord sequences. *Journal of New Music Research*, 2018. To appear.

[30] E. Waite. Generating long-term structure in songs and stories. https://magenta.tensorflow.org/2016/07/15/lookback-rnn-attention-rnn.

[31] L. C. Yang, S. Y. Chou, and Y. H. Yang. MidiNet: A convolutional generative adversarial network for symbolic-domain music generation. In *ISMIR*, pages 324–331, 2017.