Journal of Information Processing Vol.28 258–266 (Apr. 2020)

[DOI: 10.2197/ipsjjip.28.258]

Regular Paper

A Parameterized Harmony Model for Automatic Music Completion

Christoph M. Wilk[1,a)] Shigeki Sagayama[2,b)]

Received: July 7, 2019, Accepted: January 16, 2020

Abstract: In this paper, we propose harmony generation according to user input parameters based on fundamental harmonic properties as a new approach to the problem of automatic music completion (the automatic generation of music pieces from any incomplete fragments of music), which we have proposed as a generalization of conventional music information problems such as automatic melody generation and harmonization. The goal is enabling possibly inexperienced users to turn partial musical ideas into complete pieces for quick exploration of musical possibilities. Therefore, the focus lies on response to intuitive modes of input, allowing the user to intentionally shape the generated music. To that end, parameterized harmony generation utilizes fundamental musical principles which are understandable by both user and computer, instead of conventional probabilistic models (which imply imitation of a style or data corpus) or restrictive rule-based models. We apply this approach to the automatic completion of four-part chorales, using the harmonic concepts of active tones, cadences and key modulation. We implemented a system that jointly optimizes harmony and voicing considering both user input and music theory. Our system was evaluated by a professional composer and in a subjective evaluation experiment. We also invite the reader to use our system at http://160.16.202.131/automatic_music_completion.

Keywords: automatic music completion, algorithmic composition, harmony, key modulation, voicing

1. Introduction

Algorithmic composition is a popular application of artificial intelligence in music since the 1950s [1], and many algorithms have been proposed over the years [2], [3], [4]. However, many of these algorithms autonomously compose music, effectively replacing a human composer. On the contrary, our goal is to support human creativity, make music composition more accessible to users without extensive musical knowledge, and to provide tools to efficiently explore musical ideas such as melodic or harmonic motifs. This implies that it is important to meaningfully process user input. Several existing systems do this, but with significant limits to the type of input. A popular problem is constrained melody generation. Possible modes of input include lyrics and harmony progressions [5], abstract parameters [6], complete melodies to generate a countermelody [7], or melody fragments to interpolate [8] or transform [9]. The inverse problem of melody harmonization is also popular and usually processes a single, complete melody as input [10], [11], [12].

However, automatic music completion aims to go further, and simultaneously provide as many modes of input as possible, while not necessarily requiring any, i.e., an ideal system could process any amount of input in multiple domains (pitch, rhythm, voice, harmony, etc.), however incomplete, and automatically generate all missing parts in all domains. In this paper, we apply this approach to four-part chorales. We allow users to freely input notes in all four voices and also constrain harmony directly, and indirectly using the parameters proposed in this paper. An exemplary result is shown in **Fig. 1**. We chose four-part chorales, because they are a well-studied and complex discipline of classical music, with strict guidelines that facilitate the evaluation of our algorithm, but the principle of automatic music completion is applicable to any music genre. The FlowComposer [13] is a system with a similar goal to ours, and generates single melodies with associated harmony progressions while processing constraints in both domains. While not their focus, the algorithms DeepBach [14] and Coconet [15] can process note input in multiple voices to generate chorales, but do not provide the user with the means to influence the harmony, nor to deviate from the style of Bach.

In this paper, we make a case for parameterized generation of harmony progressions. Conventional approaches are usually either based on rule sets [16], [17], [18] or utilize probabilistic models, i.e., the algorithm outputs harmonies according to their prevalence in data [19], [20], [21] or according to their fitness with respect to penalty functions [22], [23]. However, such algorithms provide little freedom to influence the harmony generation process (sometimes, overall styles can be chosen). Instead, we propose to model harmony based on fundamental harmonic properties that allow users to tune the harmony generation throughout the piece. For example, the harmony could increase in complexity, culminate in a resolution, or modulate into another key wherever the user wants it to. As a first application of this principle, we present three harmony parameters, which influence how the harmony develops at every bar in the piece.

[1] Graduate School of Advanced Mathematical Sciences, Meiji University, Nakano, Tokyo 164–8525, Japan
[2] The University of Tokyo, Bunkyo, Tokyo 113–8654, Japan
[a)] wilk@meiji.ac.jp
[b)] sagayama@74.alumni.u-tokyo.ac.jp

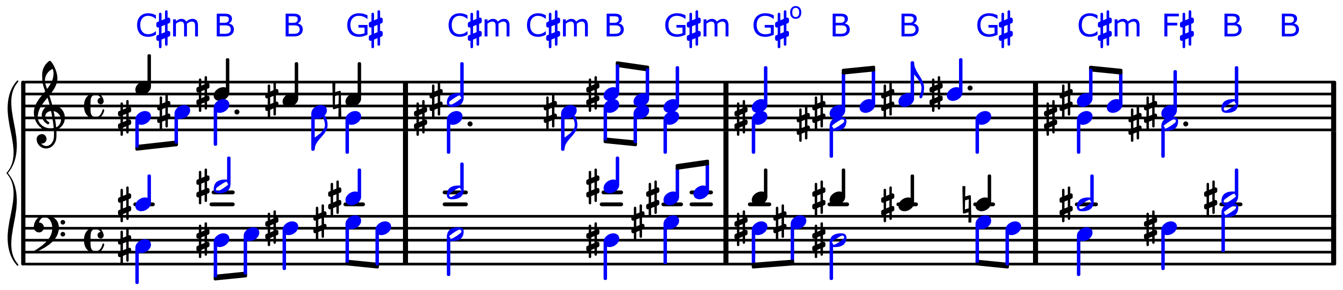


Fig. 1 A result of the automatic music completion system (input notes black, generated notes blue).

2. Automatic Music Completion

2.1 Free User Input & Parameterized Harmony Generation

Our motivation is to provide automatic music composition assistance with as much freedom to input musical ideas as possible, which is mainly influenced by two factors.

(1) Allowing input of any size: No input at all, an almost complete melody with a short section missing, or multiple melody fragments in possibly different voices.

(2) Providing multiple modes of input, e.g., pitch, harmony, rhythm, structure, or tuning parameters that allow a user to intuitively influence the automatic generation process.

The harmony model proposed in this paper addresses the second factor by providing the user with tuning parameters that determine what kind of harmony progression will be generated. While this model is applied to chorales in this paper, it is based on fundamental musical principles, and therefore also applicable to other music genres.

The GUI of our system allows users to freely input notes for four voices in piano roll format. For example, a user might think of an interesting melodic motif and insert it such that it moves through different voices (e.g., in succession from bass to soprano as in famous fugues). Our system can then compute the missing notes to turn the creative idea of the user into something complete. Furthermore, the presented harmony parameterization provides an abstract method to influence how the underlying harmony is generated. Using the parameter sliders of the interface, one can, for example, design a harmonic development that first increases in harmonic complexity, then modulates and finally resolves in the new key. While note input is quite easy for any user, the parameters are closely linked to musical concepts, and therefore, knowledge about these concepts is required to use the parameters to their full potential.

2.2 Optimization Approach

We follow the design principle of generating music pieces that best fit the user input. Therefore, we chose optimization instead of random sampling as the fundamental method for automatic composition. Our four-part chorale model is based on the common assumption [24], [25] that harmonies are hidden states of observable voicings (arrangements of notes according to the underlying harmony). Accounting for the mutual dependence of harmonic and melodic development [26], we jointly optimize the harmony progression $H \equiv (h_1, \ldots, h_N)$ and the corresponding voicing sequence $V \equiv (v_1, \ldots, v_N)$ using the following optimization objective.

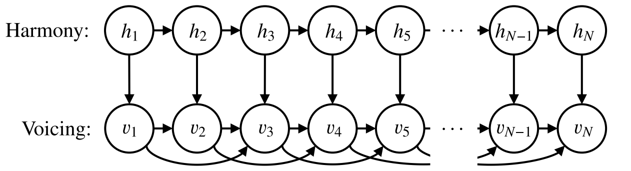

Harmony:
Voicing:
h_1 h_2 h_3 h_4 h_5 ··· h_{N-1} h_N
v_1 v_2 v_3 v_4 v_5 ··· v_{N-1} v_N

Fig. 2 A graphical representation of our music model. Harmonies h_i are regarded as hidden states of observable voicings v_i. In contrast to the typical hidden Markov model structure, voicings are also dependent on previous voicings in order to generate smooth voice leading.

$$H^*, V^* = \arg\max_{H,V} \prod_i F(v_i, h_i \mid v_{i-1}, h_{i-1}, v_{i-2}) \tag{1}$$

Note, that F does not encode a probability, but instead a weight that corresponds to how preferable a certain harmony or voicing is in the given context. For a complete definition of F, we refer to Eq. (14). However, the dependencies between harmonies and voicings in F are already illustrated in **Fig. 2**.

3. Harmony Model

3.1 General Approach

Our model is based on functional harmony, and explicitly avoids the learning of harmony probabilities from data, because optimizing according to these probabilities would lead to the generation of very common, and thus not very interesting harmony progressions. Instead, we want to enable users to decide what kind of harmony progression they want by using the parameters discussed in the next section. However, we first restrict the huge search space by constraining which transitions of functional harmonies are allowed. We denote the degree of a harmony h_i (i.e., the position of its root in the scale of a musical key) as $d_i \in \{\text{I,II,III,IV,V,VI,VII}\}$. To decide which transitions between functional degrees are acceptable, we use the following binary weight function F_D, which is obtained by analyzing a corpus of music pieces with functional harmony annotation [27].

$$F_D(d_i \mid d_{i-1}) = \begin{cases} 1 & \text{if } P(d_i \mid d_{i-1}) > T \\ 0 & \text{else} \end{cases} \tag{2}$$

where T is a threshold that determines which level of prevalence is high enough for the transition from d_{i-1} to d_i to be generally acceptable. High values of T restrict the possible transitions to more common patterns and speed up computation time, whereas low values increase the flexibility of the model at the cost of increased computation time and the risk of allowing unorthodox harmony progressions. We chose $T = 5\%$ for our experiments.

In order to generate consistent harmonic rhythms and prevent harmonies from continuing for too long, we reduce the weight of harmony continuation (i.e., $F_D(d_i \,|\, d_{i-1})$ for $d_i = d_{i-1}$) based on the strength of the current beat b_i (in 4/4 time signature). Thus, the initial weight of a harmony h_i depends on its functional degree d_i and beat b_i as follows.

$$F_H(h_i \,|\, h_{i-1}, b_i) = \begin{cases} F_D(d_i \,|\, d_{i-1}) & \text{if } d_i \neq d_{i-1} \\ 0.75 & \text{else, if } b_i \in \{2,4\} \text{ (weak beat)} \\ 0.5 & \text{else, if } b_i = \{1,3\} \text{ (strong beat)} \end{cases} \quad (3)$$

These values were chosen heuristically such that usually both one and two beat long harmonies are generated. Furthermore, the algorithm suppresses 3rd and 4th chord inversions (e.g., by 10^{-3}), since they should be avoided according to music theory.

3.2 User Input Parameters

The core of our new harmony model are parameters that can be tuned by the user and vary throughout the music piece. Our goal is to model harmony based on fundamental principles that can be understood by both user and computer, and influenced by above mentioned parameters to intentionally shape the generated harmony progressions. Therefore, we designed our model such that the relative weight of transitions between different harmonies depends only on user input.

In the following sections, we discuss several quantifiable properties $x_{\text{out}}(h_i)$ of generated output harmonies. In order to respond to user input in the form of the desired property values x_{in} (parameters), we define the following parameter weight function F_P.

$$F_P(x_{\text{out}}(h_i) \,|\, x_{\text{in}}) = 2^{-(x_{\text{out}}(h_i)-x_{\text{in}})^2/\sigma^2} \quad (4)$$

This weight is the highest if the property x_{out} of the harmony h_i equals exactly the value x_{in} desired by the user, and decreases with the difference between x_{out} and x_{in}. The tuning parameter σ determines how strongly the algorithm tries to fulfil the user's demands. At a difference of σ between x_{out} and x_{in}, the weight is reduced to half its maximum value.

3.2.1 Active Tones

In principle, we allow every chord quality (e.g., major, minor, dominant seventh, etc.) to appear on every functional degree in a harmony progression. This design choice is made based on the hypothesis that most (even uncommon) harmonies can be acceptable if they are properly resolved. In that context, the concept of active tones or tendency tones is important. These are notes that induce a desire for resolution in the listener. A famous example are leading tones, which are notes a semitone below the root note of a key, and generally exhibit a tendency, i.e., induce a desire to be resolved towards the root note. Another example are sevenths of seventh chords. Furthermore, every note not contained in the scale of a musical key can be classified as an active tone, since they exhibit the tendency to resolve back into the scale. Therefore, active tones are a useful property to compare different chord qualities. For example, a I:maj chord in a major scale contains no active tones and consequently sounds very stable, whereas a I:min7 chord in the same scale contains two active tones (the third and the seventh) and thus sounds less stable, demanding resolution. To capture this property, we define the following parameter.

$$a_{\text{out}}(h_i) = \text{Number of active tones in } h_i \quad (5)$$

This means if a user inputs $a_{\text{in}} = 1$, harmonies with one active tone have the highest weight. In our experiments, we used $\sigma = 1$ for active tones in F_P (4), i.e., for $a_{\text{in}} = 1$, harmonies with 0 or 2 active tones have half the maximum weight. Since our system jointly optimizes harmony and voicing, we can ensure that active tones are properly resolved by suppressing the weights of voicings without resolution by a factor of 10^{-6} (allowing user input to ignore this rule, but following it otherwise).

3.2.2 Cadences

The second property relates to harmonic rhythm and resolution. A few models in previous publications consider harmonic rhythm, such as the rules of the system Kulitta [21] which consider the length of harmonies, or our own previous harmony model [28]. However, these models do not guarantee consistent rhythms (e.g., important harmonies can occur on unimportant beats) and cannot be explicitly influenced by the user.

We define the next parameter based on the hypothesis that harmony progressions evoke an impression of consistent rhythm, if important cadences (harmonic resolutions) consistently occur on the same beat in a bar, which is the case in a vast number of existing music pieces. In many pieces, this dominant beat is the first beat. However, in short pieces and many of Bach's chorales, it can also be the third beat, which is what we adopt for our chorale model. To define the parameter, we classify several types of cadences based on the strength of their harmonic resolution.

- Authentic cadences, which evoke the strongest feeling of resolution and consist of a I chord without active tones and in root position, preceded by a V chord in root position.
- Imperfect cadences, into which we classify all cadences ending in a I chord in root position without active tones. This I chord can be preceded by inverted V chords, VII chords and IV chords (plagal cadence).
- Weak cadences, which comprise all movements from V, VII and IV chords to inverted I chords or VI chords (deceptive cadence). The I or VI chord can also contain active tones.

Based on these classes, we define a parameter as follows.

$$c_{\text{out}}(h_i) = \begin{cases} 3 & \text{if } h_i \text{ is I of an authentic cadence} \\ 2 & \text{if } h_i \text{ is I of an imperfect cadence} \\ 1 & \text{if } h_i \text{ is I or VI of a weak cadence} \\ 0 & \text{else} \end{cases} \quad (6)$$

We again use $\sigma = 1$ for the weight function F_P (4). However, the user input c_{in} is only considered at the third beat (the dominant beat) of each bar. On all other beats, c_{in} is set to 0 in order to avoid cadences that disrupt the harmonic rhythm. On the other hand, if a cadence occurs, the rhythm weight function F_H (3) is inverted, i.e., $F_H(h_i \,|\, h_{i-1}, b_i) = 1$ for $h_i = h_{i-1}$ and $F_H(h_i \,|\, h_{i-1}, b_i) = 0.75$ for $h_i \neq h_{i-1}$, in order to obtain longer cadential resolution chords (I or VI), which strengthen the harmonic rhythm.

3.2.3 Key Modulation

Key modulation can make harmony progressions significantly more interesting. However, many previous models either avoid its complications or use a simplistic approach. Such a common approach is the use of context-free grammars. While hierarchical rules for key modulation might be sufficient for analysis [29] or synthesis based on random sampling with low modulation probabilities [21], these rules do not capture the sequential dependency of modulations, which can lead to the problem that two neighboring chords become harmonically very distant after consecutive applications of the hierarchical modulation rule. Instead, we define the following modulation rules.

(1) Every chord in a key can be reinterpreted as a subdominant (II or IV) or dominant chord (V, rarely VII) of a new key.

(2) A consecutive modulation can only occur after a cadence has properly established the new key (which distinguishes real modulation from tonicization and modal interchange, both of which can be captured as active tones).

This fundamental principle based on reinterpretation ensures that modulations are more consistent, because the pivot chords at which the modulation occurs relate to both the previous and the following key. It even allows a wider range of modulations than hierarchical rules, because the root note of the new key does not have to be a degree of the previous key (only any subdominant or dominant degree of the new key has to be a degree of the old key). In addition, since the user can also specify how many active tones he wants thanks to the first parameter (5), he can influence how distant the new key can be (without active tones, only common chord modulation to very close keys is possible). Since modulation to a subdominant chord of the next key is generally smoother (less abrupt) thanks to the longer preparation of the new tonic, we define the modulation parameter as follows.

$$m_{\text{out}}(h_i) = \begin{cases} 3 & \text{if } h_i \text{ is part of chain modulation} \\ 2 & \text{if } h_i \text{ is dominant pivot chord} \\ 1 & \text{if } h_i \text{ is subdominant pivot chord} \\ 0 & \text{else} \end{cases} \tag{7}$$

A chain modulation is a rapid succession of key modulations, usually by fifth. We again use $\sigma = 1$ for the weight function F_P (4). If a modulation already occurred in the current bar, m_{out} of every harmony in the new key is set to 2, such that chain modulation is only preferable for $m_{\text{in}} > 2$.

3.2.4 Conflicting Parameters

There are some parameters which can have contradictory effects. For example, an authentic cadence prohibits active tones in the tonic chord. Thus, if a user inputs high parameter values for both a_{in} (demanding active tones) and c_{in} (demanding a cadence without active tones), their effects contradict each other. Therefore, a_{in} is ignored for tonic chords in such cadences, i.e., if both parameter values are high, the algorithm will try to generate an authentic cadence with many active tones in the preceding chords. Likewise, in order to facilitate modulation in cases where c_{in} is very low, i.e., a cadence might have not yet occurred in the current key, thus violating our second modulation rule defined in Section 3.2.3, c_{in} is ignored for modulation pivot chords.

4. Voicing Model

4.1 General Approach

We begin by restricting the search space using a binary constraint function F_V and then accounting for melodic context as discussed in the next sections.

$$F_V(v_i \,|\, h_i) = \begin{cases} 1 & \text{if constraints satisfied} \\ 0 & \text{else} \end{cases} \tag{8}$$

The following constraints were chosen based on music theory.

- The notes of each voice have to be within typical choir voice ranges of two octaves with the highest notes being C_6 for soprano, F_5 for alto, A_4 for tenor, and E_4 for bass.
- The notes of soprano/alto and alto/tenor have to be within the distance of a tenth from each other. Tenor and bass notes can be up to two octaves apart.
- Two nonharmonic tones are allowed per voicing. The algorithm considers passing, neighboring, and escape tones.
- Active tones cannot be doubled. On the first beat of a harmony only a non-altered fifth can be omitted.

4.2 Correction Factors

In order to generate smooth transitions between voicings, we need to account for their dependence on the preceding notes (voicings). To that end, we use what we call correction factors, which are conditional weights $F(a \,|\, b)$ learned from data. For our experiments, we used a set of 270 Bach chorales [30] for training. The correction factors have the form $F(a \,|\, b) = P(a \,|\, b)/P(a)$. The normalization using $P(a)$ removes unwanted biases towards a. For example, in case of notes, $P(a \,|\, b)$ would introduce a bias towards notes in the middle of a voice range, because these occur most often in the data. However, if a user inserts high notes, we do not want the algorithm to forcibly move the melody back to the middle of the voice range. We only want the algorithm to consider whether or not a is preferable in the current context b in order to generate good voice leading.

4.2.1 Melody Intervals

We first consider how voices move from one note to the next. Denoting a voice $x \in \{S, A, T, B\}$ (soprano, alto, tenor, bass) in a voicing $v_i = (n_i^S, n_i^A, n_i^T, n_i^B)$ comprising the four voices' notes, we define the melody interval correction factor F_I as follows.

$$F_I(n_i^x, n_{i-1}^x) = \frac{P(n_i^x \,|\, n_{i-1}^x)}{P(n_i^x)} \tag{9}$$

This factor ensures that unmelodic intervals are unlikely to be generated, and that the algorithm prefers to move voices in smaller steps, i.e., writes smoother melodies.

4.2.2 Relative Motion

The second consideration is how pairs of voices move in relation to each other. For the discussion of relative motion, we introduce the following notation for intervals $I_{i-1\to i}^{x\to y}$.

$$I_{i-1\to i}^{x\to y} \equiv n_i^y - n_{i-1}^x \tag{10}$$

The arrow is omitted for identical voices x and positions i. We define the relative motion correction factor F_R as follows.

$$F_R(n_i^x, n_i^y, n_{i-1}^x, n_{i-1}^y) = \frac{P(I_{i-1\to i}^x, I_{i-1\to i}^y \mid I_{i-1}^{x\to y})}{P(I_{i-1\to i}^x, I_{i-1\to i}^y)} \tag{11}$$

This factor encourages the generation of contrary motion, and also reduces the probability of generating forbidden parallel motion, which is, however, not completely prevented. Therefore, we additionally suppress consecutive fifths, octaves, seconds and sevenths by a factor of 10^{-6}.

4.2.3 Melodic Motion

Lastly, we consider the overall motion of melody lines. For the discussion of melodic motion, we introduce the following notation for motion types $M_{i-1\to i}^x$.

$$M_{i-1\to i}^x = \begin{cases} \text{Skip Up} & \text{if } I_{i-1\to i}^x > 2 \\ \text{Step Up} & \text{if } 0 > I_{i-1\to i}^x \geq 2 \\ \text{Hold} & \text{if } I_{i-1\to i}^x = 0 \\ \text{Step Down} & \text{if } -2 \leq I_{i-1\to i}^x < 0 \\ \text{Skip Down} & \text{if } -2 > I_{i-1\to i}^x \end{cases} \tag{12}$$

We define the melodic motion correction factor F_M as follows.

$$F_M(n_i^x, n_{i-1}^x, n_{i-2}^x) = \frac{P(M_{i-1\to i}^x \mid M_{i-2\to i-1}^x)}{P(M_{i-1\to i}^x)} \tag{13}$$

This factor reduces the probability of consecutive skips to appear in a generated melody, and lets the algorithm prefer alternating motion. Since $M_{i-1\to i}^x$ has significantly lower dimensionality than $I_{i-1\to i}^x$, we can capture larger melodic contexts without encountering data sparsity problems. In our experiments, we consider up to four previous melody notes, i.e., expand the context of F_M to $M_{i-4\to i-3}^x$. This results in the additional benefit of reducing the probability of generating consecutive unisons ($n_i^x = n_{i-1}^x$), because in the training data, voices only rarely stay on the same note for a longer time (large melodic context).

5. Optimization Algorithm

5.1 Complete Optimization Objective

Combining all elements discussed in the two previous sections, we obtain the following objective function for Eq. (1).

$$\begin{aligned} &F(v_i, h_i \mid v_{i-1}, h_{i-1}, v_{i-2}) \\ &= F_P(a_{\text{out}}(h_i) \mid a_{\text{in}})\, F_P(c_{\text{out}}(h_i) \mid c_{\text{in}})\, F_P(m_{\text{out}}(h_i) \mid m_{\text{in}}) \\ &\quad F_H(h_i \mid h_{i-1}, b_i)\, F_V(v_i \mid h_i) \prod_{x,y\in v \mid x>y} F_R(n_i^x, n_i^y, n_{i-1}^x, n_{i-1}^y) \\ &\quad \prod_{x\in v} F_I(n_i^x, n_{i-1}^x)\, F_M(n_i^x, n_{i-1}^x, n_{i-2}^x) \end{aligned} \tag{14}$$

where $x, y \in v \mid x > y$ denotes all unique pairs of voices in a voicing.

5.2 Nested Beam Search

The number of possible harmony and note combinations is very large. We have previously used Dijkstra's algorithm to explore the search space [28], which, however, is infeasible for the presented model. We have improved the computation speed by implementing a beam search algorithm [31], but this still has the drawback that sometimes certain harmonies become too dominant. For example, in a section with few constraints, almost all voicing candidates remaining in the beam can belong to a single harmony. In this case, if the beam search reaches a time step with input constraints that do not fit this harmony, the algorithm often cannot find a smooth harmony transition or voice leading.

Therefore, we have implemented a nested beam search algorithm, with a beam width for harmony w_h and for voicing w_v. This algorithm retains up to w_h different harmony candidates in the beam at each time step, and up to w_v voicing possibilities for each harmony. Ideal values for these beam widths strongly depend on user input. If set to low numbers, computation speed increases, but the system might be unable to respond well to certain input constraints, because the beam search can encounter time steps where no remaining candidate fits the user input. As initial default values in our experiments, we used $w_h = 100$ and $w_v = 10$.

5.2.1 Harmony Filtering

For beam search to be effective, the number of dead ends, i.e., harmonies without possible transition given the constraints of the next time step, should be minimized. Therefore, we apply a filtering algorithm that searches for dead ends and removes them from the search space before starting the beam search. It does this by starting at the last time step N of the piece and removing all harmony candidates that conflict with input notes at this time step. The algorithm then continues to the previous time step $N-1$ and removes all harmonies that cannot transition to any harmony remaining for time step N. From the remaining harmonies at time step $N-1$, it again removes all those that conflict with input notes at this time step, and then iteratively continues this procedure until the beginning of the piece is reached.

5.2.2 Dynamic Voice Limits

Considering that small melody steps are much more likely than large jumps, it is often unnecessary for the beam search to explore possible voicings including such jumps. However, for certain inputs (distant input notes in quick succession), these possibilities cannot be ignored. Therefore, we implemented dynamic voice limits, which restrict the search space according to user input. For each voice x at each time step i, our algorithm searches for the closest input notes of each voice before and after time step i. Based on these notes, the highest note $\hat{n}_i^x$ and lowest note $\check{n}_i^x$ that are explored at time step i for voice x are computed as follows, denoting the time steps of the closest notes of voice y before and after time step i as b_y and a_y, respectively (implying $b_y < i < a_y$).

$$\hat{n}_i^x = \sum_{y\in v} \left(\frac{n_{b_y}^y + \hat{d}_{xy}}{i - b_y} + \frac{n_{a_y}^y + \hat{d}_{xy}}{a_y - i} \right) \left(\sum_{y\in v} \left(\frac{1}{i - b_y} + \frac{1}{a_y - i} \right) \right)^{-1} \tag{15}$$

$$\check{n}_i^x = \sum_{y\in v} \left(\frac{n_{b_b}^y + \check{d}_{xy}}{i - b_y} + \frac{n_{a_y}^y + \check{d}_{xy}}{a_y - i} \right) \left(\sum_{y\in v} \left(\frac{1}{i - b_y} + \frac{1}{a_y - i} \right) \right)^{-1} \tag{16}$$

where $\hat{d}_{xy}$ and $\check{d}_{xy}$ are preferable minimum and maximum distances between the notes of two voices. In case no input note for voice y exists before the time step i, the corresponding time step b_y is undefined and thus the related terms $n_{b_y}^y + \hat{d}_{xy}/i - b_y$, $n_{b_y}^y + \check{d}_{xy}/i - b_y$ and $1/i - b_y$ are set to 0. The same applies to a_y if no input note in voice y exists after time step i. In case an input

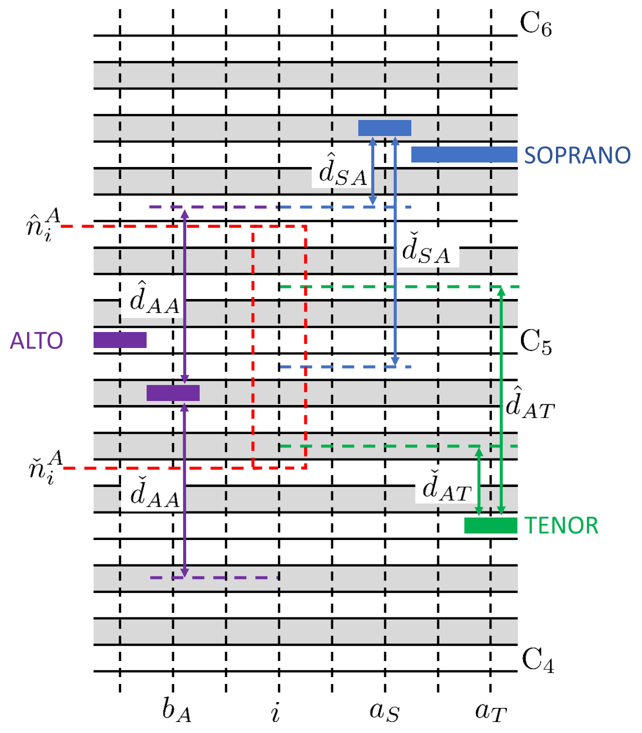


Fig. 3 Illustration of the dynamic voice limit computation shown in piano roll format for the alto voice at time step i. The dynamic limits $\check{n}_i^A$ and $\hat{n}_i^A$ restrict the search space to the notes within the red rectangle. The illustration displays the exact values of $\check{n}_i^A$ and $\hat{n}_i^A$, which are then rounded down and up, respectively. The shown example assumes that the visible notes (colored blocks) are all input notes, i.e., there are no additional input notes outside of the displayed area.

note in voice y exists exactly at time step i, this note is regarded as both $n_{b_y}^y$ and $n_{a_y}^y$, with both b_y and a_y set to 1. After computing the exact values of $\hat{n}_i^x$ and $\check{n}_i^x$, they are rounded up and down, respectively, in order to obtain integer values corresponding to actual notes. **Figure 3** illustrates a simple case, where three surrounding notes are considered. The limits $\hat{n}_i^x$ and $\check{n}_i^x$ dynamically restrict the search space to notes in preferable ranges of these input notes, which in almost all cases include the most likely candidates. Due to weighting the input notes by inverse distance from time step i, the limits are influenced the strongest by the closest input notes, ensuring that all important note candidates are considered.

$\hat{d}_{xy}$ and $\check{d}_{xy}$ were set heuristically. If x is a higher voice than y, $\hat{d}_{xy}$ denotes the maximum preferable distance between the two voices, and $\check{d}_{xy}$ denotes the minimum preferable distance. For two neighboring upper voices (S, A, T), we chose $\hat{d}_{xy} = 9$ (major sixth) and $\check{d}_{xy} = 3$ (minor third), and $\hat{d}_{TB} = 16$ (major tenth) and $\check{d}_{TB} = 5$ (fourth) for the low voices. If x is a lower voice than y, the situation is geometrically reversed, i.e., $\hat{d}_{xy} = -\check{d}_{yx}$ and $\check{d}_{xy} = -\hat{d}_{yx}$. For $x = y$, the distances specify the preferable range for melody motion, which we set to $\hat{d}_{xx} = 7$ and $\check{d}_{xx} = -7$, i.e., motion within fifths up and down. For distant voices, the values in between are summed up, e.g., $\hat{d}_{ST} = \hat{d}_{SA} + \hat{d}_{AT}$.

6. Evaluation

6.1 Experimental Setup

For the evaluation of our model, we implemented an interface that can be accessed online. The interface allows note input in eighth note resolution. However, harmonies can only change in quarter note resolution. The system computes melody contours, i.e., consecutive notes with the same pitch are automatically combined into longer notes. The harmony parameters can be set for each bar. Since objective evaluation of music is difficult, we first and foremost invite the reader to experiment with the system at http://160.16.202.131/automatic_music_completion.

6.2 Music-Theoretic Evaluation

We asked a professional composer to evaluate our system with respect to music theory. According to his assessment, the generated chorales were generally quite good and theoretically correct. In the following, we discuss problems that were still identified. As examples, we use the results shown in Fig. 1 and **Fig. 4**, which were all generated for the same input notes but different parameter values (Fig. 1) and beam widths (Fig. 4).

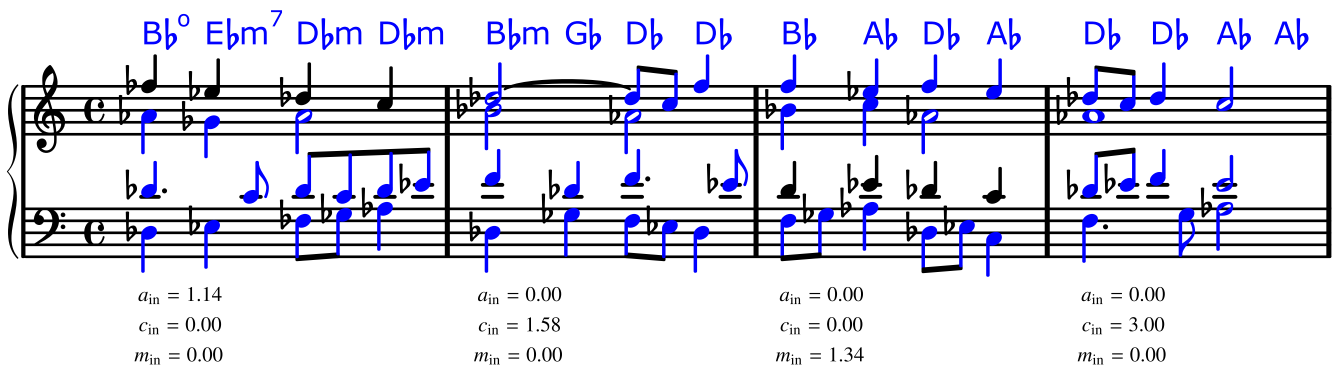
Harmonic Rhythm: The only problem in Fig. 1 is the continuation of the G# harmony (bar 2, beat 4) across the bar line, which also results in a problematic accented nonharmonic tone (F#) in the bass. This can be resolved by adjusting the weights in Eq. (3) to suppress harmony continuation across bars.

Voice Leading: The system occasionally generates problematic voice leading, e.g., the consecutive fourths in Fig. 4 (top, bar 1, between beat 3 and 4), unusual nonharmonic tones (C in the soprano, Fig. 4, top, bar 1, beat 4), or hidden parallel motion. However, the voice leading can often be improved by increasing the beam widths of the search algorithm (see Fig. 4, bottom).

Key Modulation: It can be unclear where a key modulation occurs, and the notes that change between the keys might not be properly approached by semitone. This confusion can be caused by active tones that occur during the key modulation, which reduce the clarity of the keys. However, Fig. 4 displays a case where the key modulation is unclear even without active tones. From the system's perspective, the key changes from D♭ major to A♭ major in bar 3, beat 3, but the note that changes between these keys (G♭ → G) only appears in bar 4. Furthermore, the cadence in the new key is only plagal, because $a_{\text{in}} = 0$ prevents the V chord, which contains the leading tone as an active tone. Therefore, it is unclear where or even if a key modulation has occurred. This problem cannot be resolved by increasing the beam width as demonstrated in Fig. 4. It might be resolved by treating notes that change between keys as active tones, requiring proper resolution. This would enforce that these notes appear, thus clearly identifying the key modulation. Furthermore, since these notes would increase the active tone count, the probability of other active tones appearing would decrease. Lastly, the plagal cadence might require special treatment in the context of key modulation.

6.3 Subjective Evaluation Experiment

In contrast to Section 6.2, the goal of this experiment was to obtain feedback based on musical intuition instead of theory, and 10 out of 12 participants stated to have little to no knowledge of music theory. The experiment was conducted online using the interface mentioned in Section 6.1. Participants were asked to insert some melody notes as well as use the parameters, and then evaluate the generated music according to the following criteria.



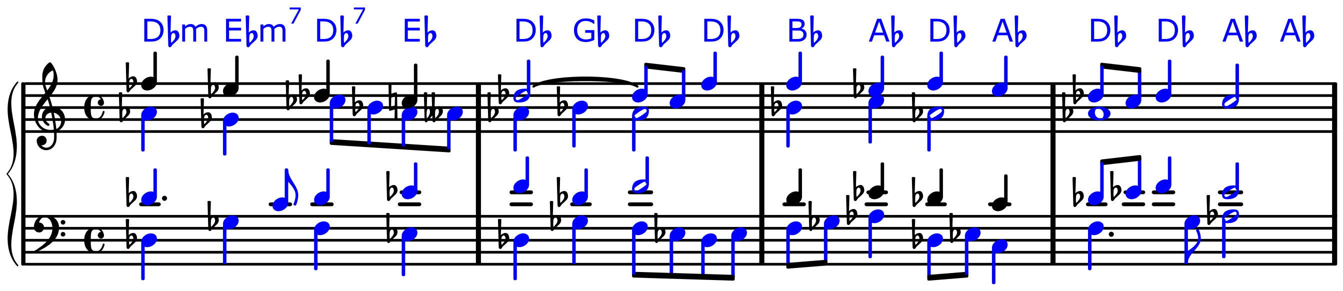


Fig. 4 Results (input notes black, generated notes blue) for the same user input (notes and harmony parameters) but different beam widths. Top: Harmony beam width = 30, Voicing beam width = 10. Bottom: Harmony beam width = 50, Voicing beam width = 20.

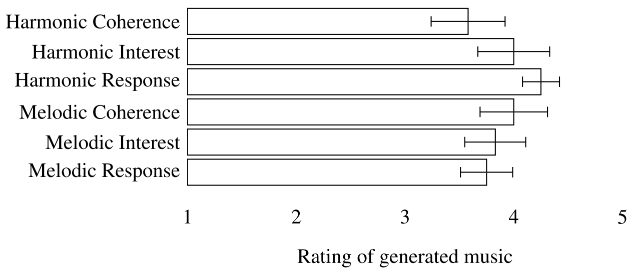


Fig. 5 The results of our subjective evaluation experiment, averaged over 12 participants. Generated music was rated with respect to the criteria explained in Section 6.3, where 1 is the worst and 5 the best rating.

- Harmonic/Melodic Coherence: How intuitively valid the generated harmonies or melodies are. Anything that contradicts the participants intuition, such as dissonance or unexpected melody jumps, decreases coherence.
- Harmonic/Melodic Interest: How interesting the generated harmonies or melodies are.
- Harmonic Response: How well the system responds to the harmony parameters, and whether the parameters are useful.
- Melodic Response: How well generated melodies go along with the input melodies.

As can be seen in **Fig. 5**, we received overall very positive feedback. Harmonic coherence was rated the worst, albeit overall still rated positively. Analysis of the generated music pieces indicates two possible reasons for the lower rating of harmonic coherence. One reason is the possible lack of music-theoretic knowledge. Large values of a_{in} and m_{in} can cause frequent occurrence of notes not contained in the current key, entailing deterioration of key coherence, which resulted in worse rating by participants who did not understand the parameters very well. However, based on feedback of 6 participants who provided comments after trying varying inputs, even users who did not know the musical concepts behind the parameters could develop an intuition for them. Thanks to this intuition, key deterioration did not have a bad effect on rating, because these users could intentionally avoid it. In general, moderate values of a_{in} and m_{in} were reported to make the harmony more interesting, while high values resulted in less pleasant results. The second reason for lower harmonic coherence is the missing consideration of psychoacoustic consonance in our model. To a certain extent, our parameters influence how consonant a harmony sounds in its context. However, in the key of C major, for example, our model treats the harmonies D major and D diminished as equal with respect to the number of active tones, although the latter would generally be perceived to be less consonant by listeners used to Western tonality [32], which resulted in worse rating of pieces with multiple diminished chords.

6.4 Quantitative Parameter Analysis

We conducted several experiments to quantitatively investigate how each parameter affects the generated harmony progressions. To do so, we generated user input constraints by randomly extracting short passages spanning two bars from Bach's chorales and randomly removing 50% of the notes in these passages. Our system then regenerated the missing notes based on varying harmony parameter values. Two types of experiments were conducted. In the first type, we further removed all notes in the second bar of the passages, such that the algorithm had more freedom to generate harmonies according to the parameter values. In the second type, the notes in this second bar were not removed, implying that is was more difficult for the algorithm to follow the harmony parameters due to stonger constraints. In both types, this second bar was quantitatively analyzed. For each parameter value, the automatic completion was repeated on 10 different random passages and results averaged.

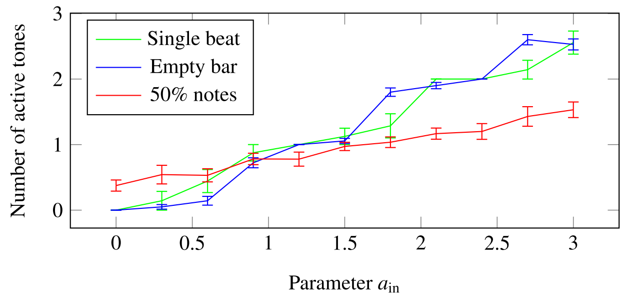


Fig. 6 Effect of the active tone parameter a_{in} ($\sigma = 1$). In an empty bar without input notes (third beat of an empty bar in green, average over whole bar in blue), the correlation with the number of active tones is quite strong. In the case of 50% input notes (averaged over one whole bar), the correlation is weaker.

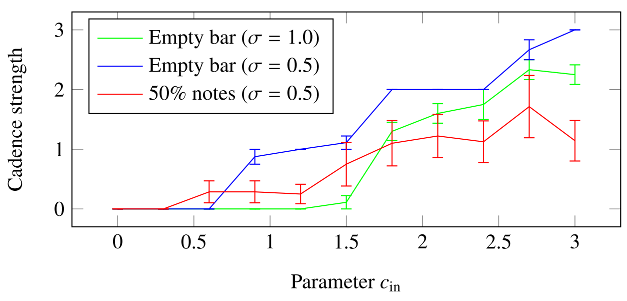


Fig. 7 Effect of the cadence parameter c_{in}. The correlation with cadence strength (from 0 = no cadence to 3 = authentic cadence, see Section 3.2.2) is relatively weak for $\sigma = 1$, and with 50% input notes especially authentic cadences are generated quite rarely.

Active Tones: For analysis of a_{in}, both c_{in} and m_{in} were set to 0. To facilitate the generation of many active tones, more chord qualities were considered: Diminished seventh, half diminished seventh, sus4 and suspended seventh chords (in addition to major, minor, diminished, dominant seventh and minor seventh chords, which were considered in all other experiments). As can be seen in **Fig. 6**, the correlation between parameter value and generated number of active tones was quite strong, when the analyzed bar did not have any input notes. If the algorithm is constrained by input notes, only a limited number of harmonic interpretations of these notes are possible, and therefore, the harmony parameter values do not translate into generated active tones as well. For example, some input notes require interpretation as active tones, such that active tones are not completely suppressed even for $a_{\text{in}} = 0$. Furthermore, Bach didn't use many exotic harmonies, causing the random passages to prevent the generation of harmonies with unusually many active tones.

Cadences: Since most cadences require active tones in the dominant chord, we chose $a_{\text{in}} = 1.5$ and set m_{in} to 0 when analyzing c_{in}. Experiments with an empty bar indicated that a value of $\sigma = 0.5$ results in significantly improved correlation between parameter value and cadence strength, as can be seen in **Fig. 7**. Regarding cadences, the input note constraints make it more difficult for the algorithm to follow parameter values compare to the active tone parameter. Since cadences require certain patterns of harmonies, input notes can prohibit such a harmonic interpretation. In particular, the authentic cadence requires a downward motion by fifth in the bass, which can be blocked by input notes. This results in generally lower values of cadence strength and significantly higher statistical uncertainty.

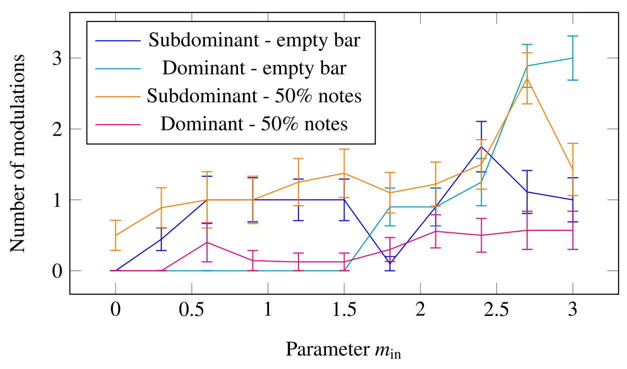


Fig. 8 Effect of the modulation parameter m_{in} ($\sigma = 1$). We distinguish between subdominant and dominant modulations. The sum of both types is the total number of modulations within the analyzed bar, e.g., this sum is 4 at $m_{\text{in}} = 3$, meaning that the harmony modulates at every beat (chain modulation).

Modulation: Modulations also often require active tones, hence we again chose $a_{\text{in}} = 1.5$ and set c_{in} to 0 when analyzing m_{in}. According to design, with increasing m_{in} one expects to first observe subdominant modulation, then a change towards dominant modulation, followed by an increase of both types (especially dominant), which can be observed well in case of an empty bar in **Fig. 8**. However, input notes make it difficult for the algorithm to follow the parameter values, and the random constraints based on Bach's chorales seem to favor harmonic interpretations with subdominant modulation. The dependence on input can be seen especially well for $m_{\text{in}} = 3$, where multiple random passages forced the algorithm to continue the same harmony for more than one beat, breaking the chain modulation (causing an unexpected decrease of the sum of the orange and magenta curves).

7. Conclusion

In this paper, we proposed a novel model for harmony progressions in order to tackle the problem of automatic music completion. It is part of a system that allows users to freely insert melodic fragments in up to four voices as input constraints, and to influence the harmony generation using tuning parameters. The system outputs complete four-part chorales. It received favorable feedback from a professional composer, and was generally well received in a subjective evaluation experiment.

The system could be improved by investigating the relation between parameters, e.g., active tones during key modulation, or the addition of a parameter for psychoacoustic consonance. Furthermore, the current parameters are closely related to concepts of music theory and require significant knowledge to display their full potential. Therefore, an additional layer of abstraction such as parameters for emotions, and investigation how they relate to the discussed musical concepts could make the system easier to use for users with little knowledge of music theory.

Our goal is to apply the principle of automatic music completion to a wider variety of music. Since the idea is not constrained to chorales, future research could realize systems that allow users to create their own jazz or pop songs, or even orchestral pieces. The focus of such systems is to realize the musical ideas of their users, making music composition more accessible to beginners as well as providing useful tools for experienced musicians.

Acknowledgments We thank Professor Yoshihiro Kanno for his competent music-theoretical analysis of our generated music.

This work was in part supported by JSPS KAKENHI Grant Number 17H00749.

References

[1] Hiller, L.A. and Isaacson, L.M.: *Experimental music: Composition with an electronic computer* (1959).
[2] Fernández, J.D. and Vico, F.: AI methods in algorithmic composition: A comprehensive survey, *Journal of Artificial Intelligence Research*, Vol.48, pp.513–582 (2013).
[3] Cope, D.: Experiments in musical intelligence, *Proc. International Computer Music Conference* (1987).
[4] Quintana, C.S., Arcas, F.M., Molina, D.A., Rodríguez, J.D.F. and Vico, F.J.: Melomics: A case-study of AI in Spain, *AI Magazine*, Vol.34, No.3, pp.99–103 (2013).
[5] Fukayama, S., Nakatsuma, K., Sako, S., Nishimoto, T. and Sagayama, S.: Automatic song composition from the lyrics exploiting prosody of the Japanese language, *Proc. 7th Sound and Music Computing Conference* (*SMC*), pp.299–302 (2010).
[6] Roig, C., Tardón, L.J., Barbancho, I. and Barbancho, A.M.: Automatic melody composition based on a probabilistic model of music style and harmonic rules, *Knowledge-Based Systems*, Vol.71, pp.419–434 (2014).
[7] Biles, J.A. et al.: GenJam: A genetic algorithm for generating jazz solos, *ICMC*, Vol.94, pp.131–137 (1994).
[8] Hirai, T. and Sawada, S.: Melody2Vec: Distributed Representations of Melodic Phrases based on Melody Segmentation, *Journal of Information Processing*, Vol.27, pp.278–286 (2019).
[9] Roberts, A., Engel, J., Raffel, C., Hawthorne, C. and Eck, D.: A Hierarchical Latent Vector Model for Learning Long-Term Structure in Music, *CoRR*, Vol.abs/1803.05428 (2018) (online), available from ⟨http://arxiv.org/abs/1803.05428⟩.
[10] Freitas, A. and Guimaraes, F.: Melody harmonization in evolutionary music using multiobjective genetic algorithms, *Proc. Sound and Music Computing Conference* (2011).
[11] Raczyński, S.A., Fukayama, S. and Vincent, E.: Melody harmonization with interpolated probabilistic models, *Journal of New Music Research*, Vol.42, No.3, pp.223–235 (2013).
[12] Pachet, F. and Roy, P.: Musical harmonization with constraints: A survey, *Constraints*, Vol.6, No.1, pp.7–19 (2001).
[13] Papadopoulos, A., Roy, P. and Pachet, F.: Assisted lead sheet composition using flowcomposer, *International Conference on Principles and Practice of Constraint Programming*, pp.769–785, Springer (2016).
[14] Hadjeres, G., Pachet, F. and Nielsen, F.: DeepBach: A Steerable Model for Bach chorales generation, arXiv preprint arXiv:1612.01010 (2016).
[15] Huang, C.-Z.A., Cooijmans, T., Roberts, A., Courville, A. and Eck, D.: Counterpoint by convolution, *ISMIR* (2017).
[16] Ebcioğlu, K.: An expert system for harmonizing chorales in the style of JS Bach, *The Journal of Logic Programming*, Vol.8, No.1-2, pp.145–185 (1990).
[17] Koops, H.V., Magalhaes, J.P. and De Haas, W.B.: A functional approach to automatic melody harmonisation, *Proc. 1st ACM SIGPLAN Workshop on Functional Art, Music, Modeling & Design*, pp.47–58, ACM (2013).
[18] Navarro, M., Caetano, M., Bernardes, G., de Castro, L.N. and Corchado, J.M.: Automatic generation of chord progressions with an artificial immune system, *International Conference on Evolutionary and Biologically Inspired Music and Art*, pp.175–186, Springer (2015).
[19] Ponsford, D., Wiggins, G. and Mellish, C.: Statistical learning of harmonic movement, *Journal of New Music Research*, Vol.28, No.2, pp.150–177 (1999).
[20] Eigenfeldt, A. and Pasquier, P.: Realtime generation of harmonic progressions using controlled Markov selection, *Proc. ICCC-X-Computational Creativity Conference*, pp.16–25 (2010).
[21] Quick, D.: *Kulitta: A framework for automated music composition*, Yale University (2014).
[22] Phon-Amnuaisuk, S. and Wiggins, G.A.: The four-part harmonisation problem: A comparison between genetic algorithms and a rule-based system, *Proc. AISB '99 Symposium on Musical Creativity*, pp.28–34, AISB London (1999).
[23] Donnelly, P. and Sheppard, J.: Evolving four-part harmony using genetic algorithms, *European Conference on the Applications of Evolutionary Computation*, pp.273–282, Springer (2011).
[24] Papadopoulos, H. and Peeters, G.: Large-Scale Study of Chord Estimation Algorithms Based on Chroma Representation and HMM, *International Workshop on Content-Based Multimedia Indexing*, *CBMI '07*, pp.53–60 (2007).
[25] Ueda, Y., Uchiyama, Y., Nishimoto, T., Ono, N. and Sagayama, S.: HMM-based approach for automatic chord detection using refined acoustic features, *2010 IEEE International Conference on Acoustics Speech and Signal Processing* (*ICASSP*), pp.5518–5521, IEEE (2010).
[26] Marx, A.B.: *Theory and practice of musical composition*, Gordon (1866). Translated by Saroni, H.S.
[27] Kaneko, H., Kawakami, D. and Sagayama, S.: Functional harmony annotation data-base for statistical music analysis, *ISMIR* (2010).
[28] Wilk, C.M. and Sagayama, S.: Harmony and voicing interpolation for automatic music composition assistance, *APSIPA* (2018).
[29] Rohrmeier, M.: Towards a generative syntax of tonal harmony, *Journal of Mathematics and Music*, Vol.5, No.1, pp.35–53 (2011).
[30] Classical Archives LLC, available from ⟨https://www.classicalarchives.com⟩ (accessed 2018-05-23).
[31] Wilk, C.M. and Sagayama, S.: Automatic Music Completion Based on Joint Optimization of Harmony Progression and Voicing, *Journal of Information Processing*, Vol.27, pp.693–700 (2019).
[32] Fujisawa, T., Cook, N.D., Nagata, N. and Katayose, H.: A Psychophysical Model of Chord Perception, *SIGMUS 90* (*2006-MUS-066*), pp.99–104 (2006).

Christoph M. Wilk was born in 1992. He received his M.S. degree from Heidelberg University, Germany in 2017. He is currently a Ph.D. student at Meiji University. His research interest is mathematical modeling of musical concepts.

Shigeki Sagayama was born in 1948. He received his B.E., M.S. and Ph.D. degrees from the University of Tokyo, Tokyo, Japan, in 1972, 1974, and 1998, respectively, all in mathematical engineering and information physics. After spending 24 years with NTT Laboratories in Tokyo and Yokosuka, Japan, and ATR Interpreting Telephony Laboratories, Kyoto, Japan, he became a Professor of the Graduate School of Information Science, Japan Advanced Institute of Science and Technology (JAIST), Ishikawa. In 2000, he was appointed Professor at the University of Tokyo. He was a Professor of Meiji University from 2014 to 2019. His major research interests include processing, recognition and synthesis of speech, music, acoustic signals, handwriting, and images. Prof. Sagayama received the National Invention Award from the Institute of Invention of Japan in 1991, the Director General's Award from the Science and Technology Agency of Japan in 1996, and other academic awards. He is a fellow of IEICEJ, a life member of IEEE and a member of the ASJ (Acoustical Society of Japan) and IPSJ.