# Music Interpolation considering Nonharmonic Tones

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

***Abstract:*** In this paper, we make a case for the problem of interpolating music when only parts of it are given. Solving this problem allows to design music composition assistant systems in which a user can insert his musical ideas and let an algorithm fill in the rest. We present such an algorithm, which interpolates four-part chorales including the explicit harmony progression. The user of the composition assistant system can input parts of the harmony progression as well as parts of the four voices' melodies. The algorithm interpolates the missing parts based on probability maximization with respect to smoothed n-gram statistics of a music corpus. In this paper, we discuss the problem that nonharmonic tones pose when explicitly modeling the underlying harmony progression as well as possible approaches to include them in the interpolation model. We present experimental outputs of the algorithm and also evaluate the results according to principles derived from music theory.

***Keywords:*** Music Interpolation, Composition Assistance, Automatic Voicing, N-grams

## 1. Introduction

Automatic music generation has been a topic of research since the automatic composition of the Illiac suite [1] in the 1950s. A lot of different approaches have been applied in this field of research since then. For an overview over research on automatic music generation, we refer to the survey paper of Fernández and Vico [2].

A significant amount of the papers published in the field of automatic music generation present algorithms that autonomously compose music. This means that while a user might choose training data in statistical models or define rules in rule-based approaches, the user has little influence on the actual generation process. Some of these approaches have gained popularity, for instance, David Cope's Experiments in Music Intelligence [3] and the Melomics music database [4]. Nevertheless, these systems are often met with criticism, since many people do not like the idea of replacing human composers with computers.

On the other hand, algorithms that provide composition assistance in some form have also been published. A popular task for computer to tackle is the problem of automatically harmonizing a melody, either adding a harmonically fitting instrument patterns to said melody or generating polyphonic accompaniment melodies based on principles of classical music theory [5, 6, 7]. This means that the user can compose a melody and let the computer generate an harmonic accompaniment.

[1] Meiji University, Graduate School of Advanced Mathematical Sciences
[a)] wilk@meiji.ac.jp
[b)] sagayama@meiji.ac.jp

The opposite has also been a topic of research, i.e. the generation of melodies based on conditions that include harmony progressions to which the melodies have to fit [8, 9, 10]. These algorithms provide user interactivity as well, letting a user create or choose harmony patterns, while the computer completes the music by composing the melody.

Both generation of melodies from harmonies as well as the opposite have been extensively researched. However, we aim to go further by developing a combined model that can be used in both directions, and additionally handle input of both types at the same time. This allows a user to insert a wide variety of music elements. For instance, the user might come up with parts of melodies of potentially multiple voices, interesting or important parts of the harmony progression, motifs that he wants to use in different places or variations, or any combination of these ideas. The algorithm then completes the piece using the input ideas as constraints.

Such a system could be very useful for both hobby composers and even professionals. Coming up with musical ideas is both an interesting process for hobby composers as well as a core discipline of professionals. However, filling in the rest of the music can be very difficult for amateurs, especially if attempting to fulfil criteria of strict music theory, or tedious for professional composers. An algorithm that can do that work automatically could therefore be an helpful assistant in both cases, while supporting the creativity of the human user. This allows him to focus on the musical ideas that define a piece. The user can reiterate the generation process multiple times, iteratively shaping the

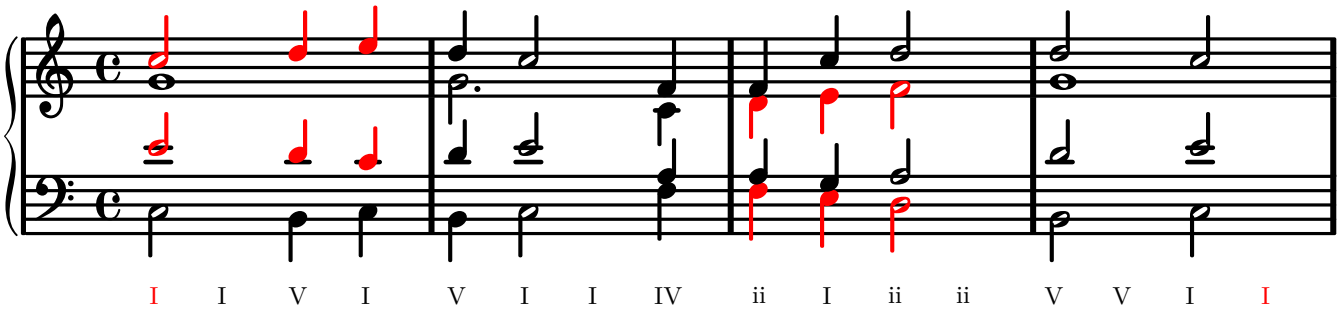


Figure 1: An exemplary result of music interpolation in the key of C major. The algorithm is constrained by partial melodies and harmony conditions, both shown in red. Minor chords are indicated with non-capital letters. The algorithm automatically connects notes within the same bar.

piece to his liking by reactively changing input conditions. Since the algorithm does the solution-oriented part of the work, which computers are suited for, the human composer can invest more time into the creative process.

The presented algorithm computes both the harmony progression as well as suitable melodies for four voices based on classical voicing principles. The approaches that to our knowledge come closest to our model are the interactive music generation systems FlowComposer [11] and DeepBach [5]. The former system is similar to ours in that it lets the user influence both harmony and melody. However, it generates lead sheets and no polyphonic melodies. The latter is a flexible harmonization model that also allows a user to input partial melodies of multiple voices, but does not explicitly account for harmony progressions.

The paper is structured as follows. In section 2, we present the basic idea of our algorithm, which interpolates four-part chorales including their harmony progression. In section 3, we discuss the problem that nonharmonic tones pose in the context of music interpolation as well as approaches to account for them by including modulation and ornamentation in our algorithm. The results of the algorithm are presented in section 4 and we conclude the paper in section 5 with a discussion on how to further expand the concept of music interpolation.

## 2. Music Interpolation

### 2.1 Chorale Completion

The algorithm presented in the following assists its user in the composition of four-part chorales. Classical chorales are music pieces for multiple voices, which are composed to have melodies that could stand on their own, while harmonizing well with each other in polyphony. A lot of music theory has been developed for this type of music, allowing to evaluate pieces objectively to a certain degree.

The presented algorithm completes four-voice pieces of which parts of melodies and the harmony progression can be input as conditions (displayed red in figure 1). The basic principle of the music generation algorithm is probability maximization, which in contrast to approaches involving random sampling outputs unique solutions for each set of constraints a user inputs. Therefore, it might be easier for the user to identify himself with the result, since less luck is involved and the output is determined by the user input. Repeating the interpolation while changing input conditions in reaction to the output of the previous iteration allows the user to intentionally shape the final result to his liking.

The interpolation process is split into the two steps of harmony progression generation and automatic voicing (assigning notes to voices) based on the resulting progression. This separation is based on the idea that harmony progressions have a certain degree of self-sufficiency as the basic structure of the music piece. A similar problem separation approach as been applied in previous research [6] and assumes a somewhat hierarchical structure of music. Aside from reduction of computational cost, the benefit of the two step approach is the possibility to use two different data sets for harmony and voicing. As data containing both four-part voicing and harmony annotation is rare, this advantage is significant and also allows to combine different genres.

### 2.2 Harmony Progression

In the first step, we interpolate the harmony progression of a piece by maximizing its probability with respect to statistics obtained from a music corpus. However, the optimization problem is constrained by conditions set by the user. Allowed or forbidden harmonies can be directly selected, and notes input by the user can further constrain the set of available harmonies at each beat.

To compute the probability of a harmony progression $H = h_1, \ldots, h_n \equiv h_1^n$, n-gram statistics are extracted from a music corpus. Using trigram probabilities, the optimization problem can be formulated as follows.

$$\begin{aligned} H_{\text{opt}} &= \arg\max_H P(H) \\ &= \arg\max_H \prod_{i=1}^{n} P(h_i | h_1^{i-1}) \\ &\approx \arg\max_H \prod_{i=1}^{n} P(h_i | h_{i-2}^{i-1}) \\ \forall h_i &: h_i \in \text{Constraints at position of } h_i \end{aligned} \tag{1}$$

In order to be able to handle unseen n-grams, which is required for certain user inputs, modified Kneser-Ney smoothing[12] is applied to the n-gram probabilities.

The harmonies $h_i$ are defined by their functional degree (I - VII), chord quality (major, minor, etc.), length in beats and onset beat, i.e. where in a bar the harmony begins. This definition allows to account for both harmonic functionality as well as harmonic rhythm, which is explained in more detail in [13]. This harmony definition is expanded to include the possibility of modulation, discussed in section 3.2.

A harmony progression can be viewed as a path through a directed graph, in which nodes are associated with harmonies and edges with multiplicative weights that correspond to the n-gram probability of the respective end nodes given the n-gram context. By multiplying the weights of all edges in a path, on can obtain the probability of the associated harmony progression. Since probabilities are by definition positive and not greater than 1, one can use Dijkstra's algorithm to compute the path with the highest probability, dynamically expanding the graph using a Fibonacci heap [14]. User input is accounted for by terminating paths at nodes that violate constraints.

### 2.3 Voicing

In the second step, we interpolate the melodies of all four voices based on statistics obtained from a music corpus as well as constraints induced by the underlying harmony progression and partial melodies inserted by the user.

The problem is formulated as probability maximization of a voicing sequence $V = v_1, \ldots, v_n \equiv v_1^n$, in which each harmony voicing $v_i = \{n_i^S, n_i^A, n_i^T, n_i^B\}$ comprises notes of the four voices soprano(S), alto(A), tenor(T) and bass(B). In the case of voicing sequences, bigram probabilities are used to compute the sequence probability, because of the large number of combinatorial possibilities.

$$\begin{aligned} V_{\text{opt}} &= \arg\max_V P(V) \\ &= \arg\max_V \prod_{i=1}^{n} P(v_i|v_1^{i-1}) \\ &\approx \arg\max_V \prod_{i=1}^{n} P(v_i|v_{i-1}) \\ \forall v_i &: v_i \in \text{Constraints at position of } v_i \end{aligned} \tag{2}$$

Furthermore, in order to utilize the power of modified Kneser-Ney smoothing [12] more effectively, a voicing probability is computed as the product of the n-gram probabilities of the notes of its four voices.

$$\begin{aligned} P(v_i) &= P(n_i^S, n_i^A, n_i^T, n_i^B) \\ &= P(n_i^S) P(n_i^A|n_i^S) P(n_i^T|n_i^A, n_i^S) P(n_i^B|n_i^T, n_i^A, n_i^S) \end{aligned} \tag{3}$$

Additionally, we approximate the probabilities to be independent from absolute pitch, for which we introduce the following formulation of intervals between notes.

$$I_i^{A\to S} \equiv n_i^S - n_i^A \qquad I_{i-1\to i}^S \equiv n_i^S - n_{i-1}^S \tag{4}$$

$$\begin{aligned} P(v_i) &= P(n_i^S, I_i^{A\to S}, I_i^{T\to A}, I_i^{B\to T}) \\ &\approx P(I_i^{A\to S}, I_i^{T\to A}, I_i^{B\to T}) \end{aligned} \tag{5}$$

Notes outside of the voices' ranges are identified during the optimization process and their probabilities set to 0. Lastly, we approximate the bigram probability of a voicing by considering the harmonies internal structure independently from the transition intervals between harmonies.

$$P(v_i|v_{i-1}) \approx \frac{1}{A} P(v_i)\, P(I_{i-1\to i}^v) \tag{6}$$

where $A$ is a normalization factor that is computed such that the probabilities sum to 1. While this approximation significantly reduces the problem of n-gram sparsity, it does not allow the algorithm to identify parallel fifths and octaves. These voice movements are to be avoided according to music theory, and their identification requires information about both internal structure (fifth interval) as well as transition properties (parallel movement). However, parallel motion is supposed to be completely avoided and a corresponding tangible rule can be easily formulated. Therefore, one can suppress parallel fifths and octaves using a penalty factor $\alpha_{\text{p}}$.

$$P(v_i|v_{i-1})^* = \alpha_{\text{p}}^{N_{\text{p}}(v_{i-1}, v_i)}\, P(v_i|v_{i-1}) \tag{7}$$

where $N_{\text{p}}(v_{i-1}, v_i)$ denotes the number of parallel octaves and fifths occurring between $v_i$ and $v_{i-1}$. Another penalty factor $\alpha_{\text{hp}}$ is used to suppress hidden parallel movement, which describes the situation where two voices move into the same direction and end up with a fifth or octave between them.

Similar to the harmony progression, Dijkstra's algorithm is used to find the optimal voicing sequence. However, the large search space is heuristically restricted by not considering voicings with the following properties.

- Voice crossings, i.e. a note in a lower voice being higher than that of a higher voice.
- Intervals between neighbouring voices that exceed the largest corresponding interval observed in the data.
- Incomplete harmonies, i.e. voicings that do not contain all pitch classes of the respective harmony.

## 3. Nonharmonic Tones

### 3.1 Problem Discussion

Music that contains notes not belonging to the current harmony pose two problems for the presented algorithm. The possibility of nonharmonic tones changes the influence of input melody notes as harmony constraints. Since an input note could be a nonharmonic tone, the number of possible harmonies significantly increases. Furthermore, during automatic voicing, the possible existence of nonharmonic tones similarly increases the size of the search space.

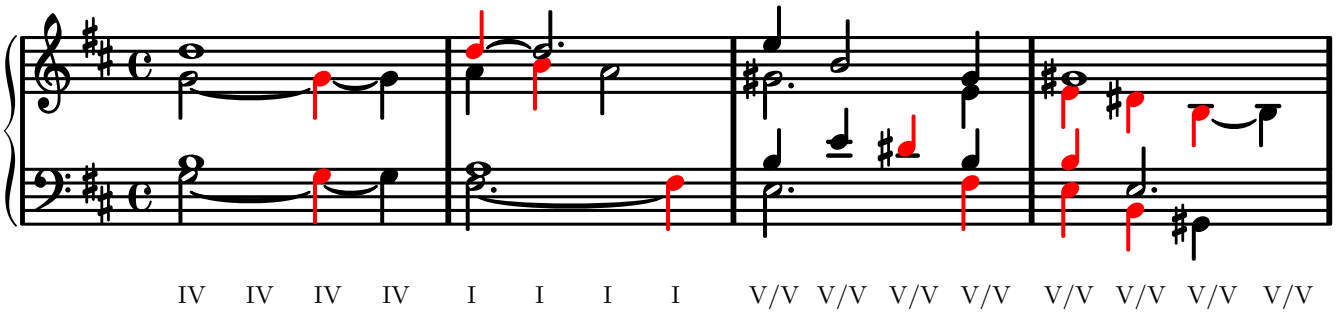


Figure 2: An example of an interpolation solution to constraints which were randomly generated from Bach excerpts (displayed in red). Both the inclusion of modulation and nonharmonic tones can be seen. As in this example, the algorithm often outputs harmony progressions containing very long and common harmonies (e.g. I and V), which can result in less interesting music due to little variation.

There are two possible reasons why notes can appear that do not belong to the expected harmony. The first is modulation, which changes the underlying key of a piece and consequently the set of available harmonies as well as available notes in the key's scale. The second is ornamentation, which utilizes notes outside of the current harmony to embellish melodies, examples being passing tones or suspension notes. The following sections detail how these concepts are incorporated in our model.

### 3.2 Modulation

To include the possibility of modulation in the model, we expand the definition of a harmony $h_i$ to include information about the current key (relative degree to basis key as well as key quality). Only relative key information is accounted for, such that for instance the following equality holds:

$$P(I \,|\, V, IV) = P(I/V \,|\, V/V, IV/V) \qquad (8)$$

Due to the large number of possible harmonies, every chord quality in the data is reduced to its most similar triad, e.g. major seventh chords are reduced to major triads.

Since only relative pitch relations between notes and modulation key's are extracted as statistical data, the algorithm is effectively able to compute the most likely key of the partial melodies input by the user, by yielding a harmony progression modulated with respect to a random base key.

### 3.3 Ornamentation

The algorithm ensures complete harmonies by treating four-voice harmonies as the collection of the three notes of the triad and an additional fourth note. The generation of the three triad notes is enforced during the voicing process. The fourth note can be a duplicate of a harmony note, but also a nonharmonic tone. Both during induction of harmony constraints from notes as well as automatic voicing, the probability of harmonies containing a nonharmonic tone is reduced using a penalty factor which differs depending on whether the note is contained in the current key's scale ($\alpha_\mathrm{s}$) or not ($\alpha_\mathrm{ns}$).

$$P(v_i)^* = \alpha_\mathrm{s}\, P(v_i) \text{ if } \exists n \in v_i : n \notin h_i \text{ and } n \in s_i \qquad (9)$$

where $s_i$ is the scale at the current index $i$. In case the note is not even contained in the scale, the smaller penalty factor $\alpha_\mathrm{ns}$ is applied.

## 4. Experimental Results

### 4.1 Training Data

For the harmony interpolation, the KSN annotation data set [15] was used as training data, which contains information about harmonic rhythm and functionality. The data contains about 9100 bars of harmony progressions.

For the voicing interpolation, Bach's chorales were obtained as training data from the Classical Archives website [16] in MIDI format. After excluding chorales with more or less than four voices, 380 pieces remain, containing about 27000 beats. The voices are read by the program in eighth note resolution, i.e. shorter note ornaments are ignored.

### 4.2 Music Theory

In music theory, there are several rules a chorale composer should generally follow. We implemented evaluation functions for the following principles.

(1) Parallel motion: Parallel fifths and octaves are to be avoided. Hidden parallels as well (but not as strictly).
(2) Inter-voice distance: The intervals between neighboring voices should have sizes within a 8th interval for higher voices, and within a 10th interval between tenor and bass.
(3) Smooth voice leading: Melody intervals should not be too large and be contained in the set of melodic intervals: Major/minor second, major/minor third, perfect fourth, perfect fifth, ascending minor sixth and perfect octave intervals). We refer to intervals outside of this set as unmelodic.

Classic composers sometimes ignore these rules, since they are rather guidelines than absolute laws. Therefore, we compared the results of our algorithms with statistics of Bach's chorales. For the evaluation, 90% of Bach's chorales were used as training data for the voicing algorithm, and from the remaining 10% of the chorales 100

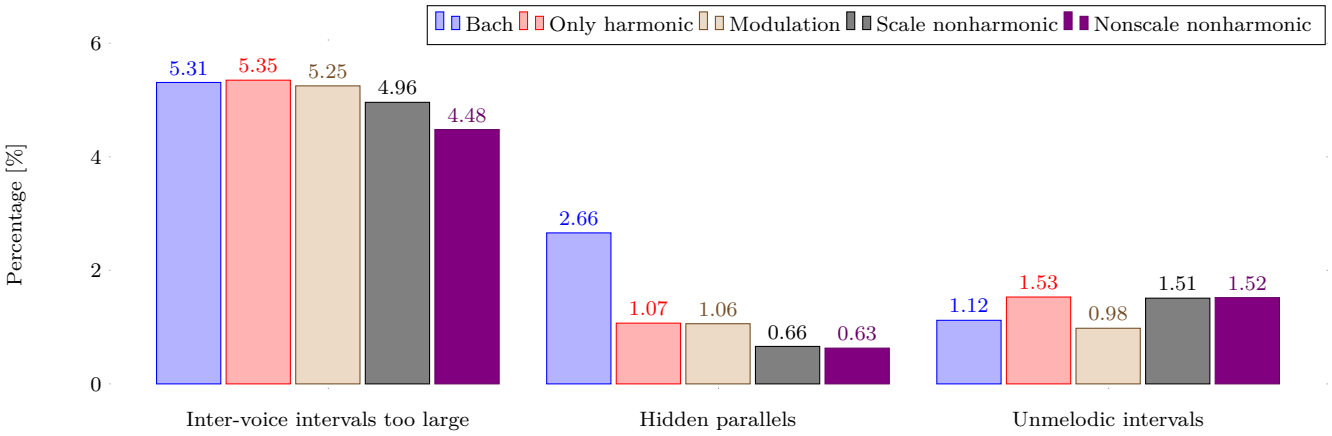


Figure 3: Music theoretic properties of the experimental results, comparing Bach's chorales' statistics with the results of algorithm outputs when not considering nonharmonic tones (Only harmonic), considering modulation (Modulation), additionally allowing nonharmonic tones withing the key's scale (Scale nonharmonic), and additionally allowing nonharmonic tones outside the scale (Nonscale nonharmonic). Data about hidden parallel motion is displayed, because parallel motion is almost completely suppressed by the algorithm.

excerpts spanning 4 bars were randomly extracted. 80% of the notes were removed from the excerpts and the resulting 20% of the notes were used as input constraints for the interpolation experiment. During preliminary test runs, the following parameter values were heuristically chosen: $\alpha_{\mathrm{p}} = 0.002, \alpha_{\mathrm{hp}} = 0.02, \alpha_{\mathrm{s}} = 0.5, \alpha_{\mathrm{ns}} = 0.1$.

The first benefit of including nonharmonic tones in the model is its increasing flexibility, allowing the algorithm to find solutions to more difficult input constraints. When suppressing parallel motion and ignoring nonharmonic tones, the algorithm did not yield solutions for 16 of the 100 excerpts. The inclusion of modulation in the harmony model reduced the number of unsolvable constraints to 12, including nonharmonic scale tones reduced it to 3, and additionally allowing nonharmonic tones outside the scale made the algorithm yield results for every excerpt.

Secondly, as can be seen in figure 3, incorporating nonharmonic tones in the music model allows the algorithm to more easily adhere to music theory concerning parallel motion suppression and inter-voice intervals. However, the same does not seem to be the case with melody intervals, where inclusion of nonharmonic tones increases the number of resulting unmelodic intervals. A more sophisticated model of nonharmonic tones might improve the results, if it considers the development of melodies additionally to voicing constraints.

### 4.3 Subjective Evaluation

While most results do not sound bad, they sometimes sound not really interesting. With the current algorithm, interesting user input is required to produce interesting results. The algorithm maximizes probability, which is good for adhering to music theoretical rules, but interesting musical twists often are not the most likely ones.

When including nonharmonic tones, the algorithm sometimes generates voicings that sound dissonant in the musical context. This is difficult to evaluate objectively, but shows that a model of consonance might be needed. Lastly, the harmonic rhythms of the two used data sets (harmony and voicing) are quite different, which results in frequent long harmonies in common progressions with few interesting developments. A method to intentionally deviate from maximum probability might improve this aspect.

## 5. Conclusions

We made a case for the problem of music interpolation and presented an algorithm that completes four-part chorales, where parts of melody and harmony can be defined by the user. This is achieved by finding the optimal sequence of harmonies and voicings according to smoothed n-gram statistics. We further discussed the problem of nonharmonic tones, and how to incorporate them into the music model, increasing the flexibility of the model, which allows the algorithm to solve more interpolation problems and adhere stronger to music theoretical rules.

The algorithm could be further improved by introducing a more powerful model of harmonic rhythm and larger harmonic structure, which would further increase flexibility and make interpolation of longer pieces more feasible. Finally, one could implement a melody model that not only makes melodies adhere to music theory, but also generate them to be more interesting, possibly using neural networks.

A major problem remains the weakness of probability maximization in the context of music resulting in correct but uninteresting results. Random sampling might occasionally yield more interesting outputs, but an even more favorable approach would be a method to intentionally deviate from maximum probability solutions.

**Acknowledgments** This work was partially supported by JSPS KAKENHI Grant Number 17H00749.

## References

[1] Lejaren Arthur Hiller and Leonard Maxwell Isaacson. Experimental music: Composition with an electronic computer. 1959.

[2] Jose D Fernández and Francisco Vico. AI methods in algorithmic composition: A comprehensive survey. *Journal of Artificial Intelligence Research*, 48:513–582, 2013.

[3] David Cope. Experiments in musical intelligence. In *Proceedings of the International Computer Music Conference. San Francisco*, 1987.

[4] Carlos Sánchez Quintana, Francisco Moreno Arcas, David Albarracín Molina, Jose David Fernández Rodríguez, and Francisco J Vico. Melomics: A case-study of AI in spain. *AI Magazine*, 34(3):99–103, 2013.

[5] Gaëtan Hadjeres, François Pachet, and Frank Nielsen. Deepbach: a steerable model for bach chorales generation. *arXiv preprint arXiv:1612.01010*, 2016.

[6] Hermann Hild, Johannes Feulner, and Wolfram Menzel. Harmonet: A neural net for harmonizing chorales in the style of js bach. In *Advances in neural information processing systems*, pages 267–274, 1992.

[7] Francois Pachet and Pierre Roy. Musical harmonization with constraints: A survey. *Constraints*, 6(1):7–19, 2001.

[8] Satoru Fukayama, Kei Nakatsuma, Shinji Sako, Takuya Nishimoto, and Shigeki Sagayama. Automatic song composition from the lyrics exploiting prosody of the japanese language. In *Proc. 7th Sound and Music Computing Conference (SMC)*, pages 299–302, 2010.

[9] John A Biles et al. Genjam: A genetic algorithm for generating jazz solos. In *ICMC*, volume 94, pages 131–137, 1994.

[10] Carles Roig, Lorenzo J Tardón, Isabel Barbancho, and Ana M Barbancho. Automatic melody composition based on a probabilistic model of music style and harmonic rules. *Knowledge-Based Systems*, 71:419–434, 2014.

[11] Alexandre Papadopoulos, Pierre Roy, and François Pachet. Assisted lead sheet composition using flowcomposer. In *International Conference on Principles and Practice of Constraint Programming*, pages 769–785. Springer, 2016.

[12] Stanley F Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 310–318. Association for Computational Linguistics, 1996.

[13] Christoph M Wilk, Souya Ito, and Shigeki Sagayama. N-gram based chord progression interpolation for the completion of partial four-part chorales. *IPSJ*, 2018.

[14] Michael L Fredman and Robert Endre Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM (JACM)*, 34(3):596–615, 1987.

[15] Hitomi Kaneko, Daisuke Kawakami, and Shigeki Sagayama. Functional harmony annotation data-base for statistical music analysis. *ISMIR*, 2010.

[16] Classical Archives LLC. https://www.classicalarchives.com. Referenced: May 23, 2018.