Harmony and Voicing Interpolation for Automatic Music Composition Assistance

Christoph M. Wilk and Shigeki Sagayama
Meiji University, Tokyo, Japan
E-mail: {wilk, sagayama}@meiji.ac.jp

***Abstract*—In this paper we make a case for automatic composition assistance centered around the creative intention of the user. After discussing how this focus on the user can influence algorithm design choices, we present a composition assistant system that follows such principles. The system allows a user to freely define any part of a four-part chorale to be composed, be it harmony or partial melodies for up to four voices. The algorithm interpolates the user input and yields a complete four-part voicing including the underlying harmony progression. The interpolation is based on n-gram statistics, to which smoothing is applied in order to enable the algorithm to handle a wide range of user inputs even if they do not appear in the training data. The system computes the solution by maximizing probability with respect to the statistics using Dijkstra's algorithm and dynamic search space expansion. The experimental results show that the composition assistant produces valid results, quite successfully adhering to rules derived from music theory.**

I. Introduction

The automatic generation of musical content has been a topic of research since the automatic composition of the Illiac suite[1] in the 1950s. Research on artificial intelligence has since come a long way, and a variety of methods have been applied to the problem of music composition, including rule-based systems, grammatical models, evolutionary algorithms and neural networks. For an overview over research on artificial intelligence in the field of music generation, we refer to the survey of Fernández and Vico[2].

Many of the published approaches to automatic music generation aim at enabling a computer to autonomously compose its own music, oftentimes imitating a certain style or genre. Some of these systems yield musically interesting results, examples being David Cope's Experiments in Music Intelligence[3] or the Melomics music database[4]. However, since these systems essentially replace a human composer, they have often been met with skepticism, all the while human composers are still regarded as superior to artificial ones.

On the other hand, automatic composition systems that include a significant amount of user interaction have received considerable public attention. One example is a system called FlowComposer[5], which interactively assists its user in lead sheet creation and has been used by the artist Skygge to compose a quite successful album[6]. Another example is a system called Orpheus[7], which provides a web-based service to turn a Japanese text into a song, and has been used by several thousand website visitors. In our research, we specifically focus on user interaction and in this paper present an algorithm for automatic music composition assistance.

This paper is structured as follows. In section II, we present the structure of our algorithm that assists a user in composing four-part chorales, and discuss how our focus on user interaction has influenced design choices. The following sections III and IV detail the algorithm. It is divided into the two steps of inferring the underlying harmony progression of a section to be composed, and then generating a four-part voicing of said progression. The results of the algorithm are discussed in section V and the paper is concluded in section VI, ending in a discussion how user centered composition assistance can be developed further.

II. Composition Assistance

A. Autonomous Composition vs. Assistance

Autonomous composition and composition assistance share common aspects such as mathematical models that allow computers to understand or learn various aspects of music. Nevertheless, the focus and problems to be solved can be quite different. Autonomous composition aims at replacing a human composer, with the ultimate goal of passing a musical Turing test of sorts. As such, algorithms of this type have to, for example, imitate a certain style of music. Most importantly, however, they have to somehow capture human creativity in order to create results that do not sound too boring or mechanical. On the other hand, composition assistance can rely on the human user to provide and judge musical ideas to a certain extent. However, in comparison with an autonomous composer, an artificial assistant is subject to far more constraints, which represent the intention of the user.

B. Completion of Partial Four-Part Chorales

In this paper, we present a composition assistance algorithm that was designed with focus on its user. It assists in composing four-part chorales, which is a major discipline of classical music. These chorales are pieces for four voices with different vocal ranges, often sung by choirs, but also performed with instruments, e.g. by string quartets. One of a chorale's main features is that voices are composed to be distinct from each other and interesting on their own, while their interplay is also important. This property has lead to a lot of music theory related to chorales, making it a popular composition problem to be solved by students or computers.

Systems to harmonize given melodies[8], [9], [10] as well as systems tackling the inverse problem of conditional melody generation[7], [11], [12] have been published, but for our

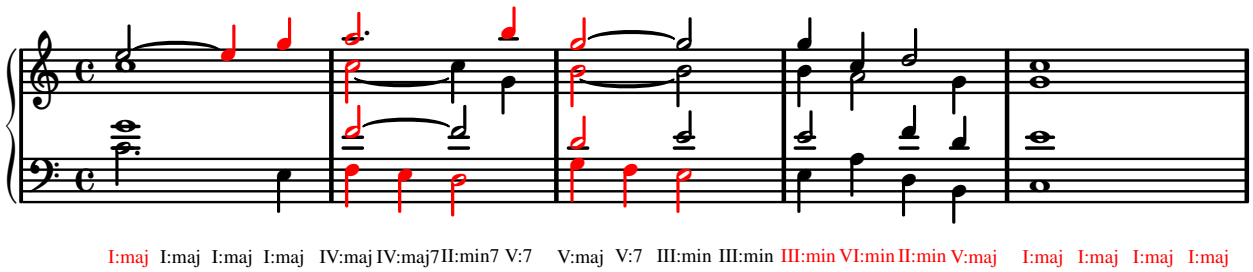

Fig. 1. An exemplary result of chord and voicing interpolation in C major. The algorithm is constrained by both partial voicing and harmony shown in red. While user constraints can be set per beat, the algorithm treats harmonies as objects with length. For example, it regards the first harmony as a tonic chord spanning four beats, which is allowed because it satisfies all constraints during its duration (harmony of the first beat and notes of third and fourth beat).

composition assistant we aim even further and allow the user to more freely insert his musical ideas, be it harmonies or notes of any voice (displayed red in Fig. 1). For example, the user might only come up with a part of a melody, a motif which might move through different voices, an interesting baseline or partial chord progression, or any combination of these ideas, and then use the composition assistant to fill in the rest of the piece. When completing the chorale, the algorithm splits the generative process into the interpolation of harmony and voicing (turning abstract harmonies into concrete notes). This two step approach is based on the idea that harmony progressions are important on their own, and similar problem separation has been used for autonomous composition in previous research[9]. Furthermore, this separation allows to use two different and independent data sets for harmony and voicing, drastically increasing the amount of usable data and the system's flexibility (e.g. one could use jazz harmonies with classical voicing). The concept of chorale completion is illustrated in Fig. 1 with an exemplary result. The algorithm was designed considering how a user can best identify himself with a creation that is in part result of artificial intelligence, which is discussed in the following.

C. User Centered: Unique Solutions

For capturing musical concepts with computers, the most popular and flexible approaches involve probabilistic models. When dealing with probabilities, there are two main principles that can be applied to generate content.

1) Maximizing the probability, corresponding to assuming the existence of rules and principles in the data, and aiming at fulfilling these to the highest possible degree.
2) Drawing random samples from the probability distribution, corresponding to generating imitations with statistical properties as similar as possible to the training data.

A recent example of successful application of the second principle is the automatic composition system called DeepBach[8], which was also developed for the classical problem of chorale composition. However, it differs from our system in that its focus is the imitation of the composer Bach to the extent that generated chorales are indistinguishable from real ones. While DeepBach is also able to harmonize melodies inserted by the user, methods involving random sampling have a drawback with respect to composition assistance, which is their inherent randomness. This means, for example, that the system generates different results for the same melody input depending on random number generator seeds. Therefore, luck is involved and the user's direct influence decreases. In a sense, when regenerating the same piece with different seeds, the human is assisting the computer by choosing its best creations, shifting the focus from user to computer. Therefore, we think that the randomness of the generation process should be minimized if possible.

On the contrary, the method of probability maximization associates every set of inputs with a unique solution. This means that a user cannot simply retry automatic generation with the same conditions, but has to intentionally change something to achieve different results. For example, he could add constraints to change the harmony flow or add notes to shape melody or bass line where it does not yet suit his taste. This in turn means that the result is shaped by the user's intention, not involving luck, possibly increasing sense of accomplishment and identification with the creation. The automatic composition system Orpheus[7] follows this approach, where users often spend a considerable amount of time tuning parameters of their songs. Often voiced feedback is that they enjoy being able to compose their own music without requiring the extensive knowledge of professionals, indicating that the users are able to view the generated songs as a creation of their own.

However, probability maximization can have the drawback, that with too few constraints the results may sound boring, because of being too common. Therefore, user input plays an important role for the output, be it melodic elements or interesting harmonies. While not yet implemented, the user could further be allowed to manually tune parameters like harmonic tension, intentionally moving probability mass from common to more uncommon patterns, which is discussed in the outlook in section VI-C.

D. Implicit vs. Explicit Information

Implicit modeling of musical aspects can increase versatility of the mathematical model, if enough data is available. For example, the system DeepBach[8] makes use of a neural

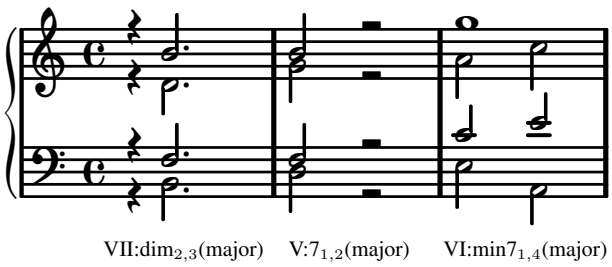

Fig. 2. Three exemplary harmonies in the key of C major (key quality in round brackets). The subscript denotes $onset, length$ in beats. As for the rightmost symbol, the algorithm treats consecutive different voicings of the same harmony as continuation of the respective harmony. Key quality and rhythm information are in this paper often omitted for readability.

network to implicitly learn harmonic relationships, allowing it to automatically account for nonharmonic tones. This means that data with harmony annotation is not required, but in turn information about the harmony is not provided by the system.

On the contrary, we chose to explicitly model the underlying harmony progressions of music pieces, because it provides the user with more ways to intuitively influence the composition of their piece.

III. Harmony Interpolation

A. Harmony Constraints

To interpolate the underlying harmony of a piece, the algorithm computes the harmony sequence with the highest probability with respect to statistics obtained from a music corpus. The user can influence the harmony progression generation with the following two types of constraints.

1) Harmony candidates can be manually specified for each beat. Harmony progressions are only valid if they contain one of those candidates at the respective beat.
2) Additional constraints are inferred from the partial voicing, i.e. notes that were input by the user. A harmony is only allowed if it contains all notes that occur on beats during its duration.

It is possible that these constraints exclude all harmonies at a beat, in which case the user is informed of the infeasibility of the interpolation problem.

Mathematically, one can formulate the problem of finding the optimal harmony sequence $H = h_1, \ldots, h_n \equiv h_1^n$ as follows.

$$H_{\text{opt}} = \arg\max_H P(H) = \arg\max_H \prod_{i=1}^{n} P(h_i|h_1^{i-1}) \quad (1)$$
$$\forall h_i : h_i \in \text{Constraints at position of } h_i$$

The shorthand notation h_a^b, denoting a sequence from index a to index b, is used throughout this section.

B. Harmony Definition

For the quality of the interpolation it is important how the harmony symbols h_i are defined. The presented algorithm accounts for both harmonic functionality and rhythm by using symbols h_i that encode the following information (see Fig. 2).

- Functional Degree: The interval between the root notes of key and harmony, denoted in roman numerals from I to VII, using ♭ accidentals for minor intervals.
- Chord Quality: For the presented results, we included major, minor, diminished, augmented, dominant seventh, major seventh and minor seventh chord qualities.
- Key Quality: This information is, for example, required to discern between V:maj harmonies in major and harmonic minor keys, which are expected to be followed by different harmonies (major and minor tonics, respectively).
- Harmony Length: Given in beats and together with the onset information allows the algorithm to produce meaningful harmony rhythms.
- Harmony Onset: A value from 1 to 4, which specifies on which beat in a bar the harmony begins.

C. Probability Computation

3-gram statistics are used to approximate the probabilities of the optimization problem in (1).

$$P(h_i|h_1^{i-1}) \approx P(h_i|h_{i-2}^{i-1}) \quad (2)$$

The n-gram method, originally developed for language processing, has been applied successfully for music analysis in previous research[13], which is often attributed to grammar-like properties observed in music. In a naive n-gram model, the 3-gram probability in (2) would correspond to the number of occurrences of the sequence h_{i-2}^i in the training data divided by the total number of sequences of the same length.

However, due to the user being able to freely constrain the harmony, the occurrence of 3-grams h_{i-2}^i that were not seen in the music corpus is quite probable, and the algorithm has to be able to handle both unseen candidates h_i as well as unseen contexts h_{i-2}^{i-1}.

The problem of unseen candidates h_i is handled using Kneser-Ney smoothing. The idea of this smoothing method is to use a discount value D to subtract probability mass from higher order n-grams and distribute it to lower order n-grams as follows.

$$P_{KN}(h_i|h_{i-n+1}^{i-1}) = \frac{c(h_{i-n+1}^i) - D}{\sum_{h_i} c(h_{i-n+1}^i)} + \gamma\, P_s(h_i|h_{i-n+2}^{i-1})) \quad (3)$$

where the function $c(h_a^b)$ counts the number of occurrences of the n-gram h_a^b in the training data, and γ is a factor that is computed such that the total probability sums to 1. The smoothing probability P_s is not a classical n-gram probability, but instead depends on the number of contexts in which the reduced n-gram appears in the training data.

$$P_s(h_i|h_{i-n+1}^{i-1}) = \frac{C(\cdot\, h_{i-n+1}^i) - D}{\sum_{h_i} C(\cdot\, h_{i-n+1}^i)} + \gamma\, P_s(h_i|h_{i-n+2}^{i-1}) \quad (4)$$

$$C(\cdot\, h_{i-n+1}^i) = |\{h'|c(h', h_{i-n+1}, \ldots, h_i) > 0\}|$$

The recursion terminates at 1-grams $P_s(h_i)$, where D and γ are set to zero. The discount value D is usually computed

depending on how much data is available for a given context according to the Good-Turing estimate.

$$D = \frac{n_1}{n_1 + 2n_2} \tag{5}$$

where n_1 and n_2 are the number of symbols which occur in the training data exactly one or two times, respectively. However, in case of music, the number of different symbols is much smaller than their average count, resulting in very small and unstable values of n_1 and n_2, possibly even zero. Therefore, we set $D = 1/3$, corresponding to the approximation $n_1 \approx n_2$. For more detailed analysis on Kneser-Ney smoothing we refer to the experimental evaluation of Chen and Goodman[14].

Finally, the described smoothing method cannot account for unseen contexts, because in that case division by zero occurs in (3). Therefore, in case a context $h_{i-n+1}^{i-1} \equiv h_c$ does not appear in the training data, the algorithm falls back to a reduced context h_c^-.

$$P(h_i|h_c) = \begin{cases} P_K N(h_i|h_c) & \text{if } \sum_{h_i} c(h_{i-n+1}^{i}) > 0 \\ P_K N(h_i|h_c^-) & \text{otherwise} \end{cases} \tag{6}$$

The simplest fallback method would be to set $h_c^- = h_{i-n+2}^{i-1}$, i.e. removing the last symbol. However, we insert intermediate fallback levels where only the rhythm information of the last symbol is removed. For example, if the context (II:min$7_{3,2}$,VI:maj$7_{1,2}$) is unseen, the algorithm first falls back to searching contexts with a VI:maj$7_{1,2}$ harmony preceded by II:min7 harmonies of any length. Only if this context is also unseen, it is reduced to the single VI:maj$_{1,2}$ harmony.

D. Probability Maximization

One can view a harmony progression as a path through a directed graph with harmony nodes. Each edge in the graph is associated with a multiplicative weight that corresponds to the n-gram probability of its end node given the preceding nodes. The total probability of a path can be obtained by multiplying all weights of its edges. Since these probabilities are all positive, one can use Dijkstra's algorithm to compute the path with the highest probability. The graph is dynamically expanded using a Fibonacci heap [15]. User input is accounted for by terminating paths at nodes that violate constraints.

IV. Voicing Interpolation

A. Problem Definition

The harmony progression obtained in the previous step is turned into actual notes by choosing chord voicings, i.e. selecting pitches for tones contained in the harmonies and assigning them to different voices. In case of four-part chorales, a 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). Internally, the notes are processed as MIDI note numbers (integers corresponding to pitch), entailing that enharmonic spelling is ignored. Similar to the harmony interpolation, the problem is formulated as constrained probability maximization of a voicing sequence $V = v_1, \ldots, v_n \equiv v_1^n$.

$$V_{\text{opt}} = \arg\max_V P(V) = \arg\max_V \prod_{i=1}^{n} P(v_i|v_1^{i-1}) \tag{7}$$
$$\forall v_i : \forall n_i \in v_i : n_i \in \text{Harmony and Partial Voicing}$$

The constraints are the underlying harmony progression, i.e. notes have to be contained in the harmony at their position, and the existing partial voicing, meaning all notes that the user has manually inserted (shown as red notes in Fig. 1).

B. Probability Computation

Analogously to the harmony interpolation, n-gram statistics are used to compute voicing probabilities, in order to benefit from the smoothing method described in section III-C. This allows the algorithm to handle user input that does not occur in the training data. However, since the number of possible voicings is much larger than the number of possible harmonies, the context size is reduced to voicing 2-grams $P(v_i|v_{i-1})$. To further increase flexibility, the voicings v_i are not directly used as symbols in the computation, but instead their probabilities are factorized into n-gram probabilities of the notes $n_i^S, n_i^A, n_i^T, n_i^B$ that make up a voicing.

$$\begin{aligned} P(v_i) &= P(n_i^S, n_i^A, n_i^T, n_i^B) \\ &= P(n_i^S)P(n_i^A|n_i^S)P(n_i^T|n_i^A, n_i^S)P(n_i^B|n_i^T, n_i^A, n_i^S) \end{aligned} \tag{8}$$

This means that note 4-grams already occur for voicing 1-grams. In order to account for transitions between voicings (i.e. 2-grams), note 8-grams would be required, which entails sparsity problems considering the relatively small size of music data sets.

For the following, we introduce the a notation for intervals between two notes.

$$I_i^{A\to S} \equiv n_i^S - n_i^A \qquad I_{i-1\to i}^S \equiv n_i^S - n_{i-1}^S \tag{9}$$

Since the absolute pitch values of two notes encode the exact same information as the pitch value of one of the notes in combination with the interval size between the two notes, the following equation holds.

$$P(v_i, v_{i-1}) = P(v_i, I_{i-1\to i}^v) \tag{10}$$

where $I_{i-1\to i}^v = \{I_{i-1\to i}^S, I_{i-1\to i}^A, I_{i-1\to i}^T, I_{i-1\to i}^B\}$ are the intervals between the consecutive notes of each of the four voices in the two voicings. For the right hand side of (10) we assume the following approximate statistical independence, which allows to express the voicing 2-gram probability as a product of 1-gram probabilities.

$$P(v_i, I_{i-1\to i}^v) \approx \frac{1}{A} P(v_i)\, P(I_{i-1\to i}^v) \tag{11}$$

where A is a normalization factor. This approximation reduces n-gram sparsity drastically and still allows the model to account for several important aspects of polyphonic voicing. From a composer's perspective, this corresponds to separately considering a voicings internal structure and the transition

between voicings. The structure probability $P(v_i)$ enables the algorithm to reproduce the following relationships between voices.

- The occurrence of chord inversions (i.e. which note appears in the bass voice) and the duplication of voices in triad harmonies.
- The psychoacoustic consonance of a voicing, which is difficult to express as tangible rules, but its significance becomes quickly apparent when listening to the two voicings of the rightmost harmony shown in Fig. 2.
- The balance between inter-voice distances, i.e. the intervals between neighbouring voices.

Likewise, the transition probability $P(I^v_{i-1\to i})$ can express the following aspects of voicings.

- The balance between step sizes of all four voices from the previous note to the next.
- The balance between parallel, oblique and contrary motions of the four voices.

While resolving the sparsity problem, the assumed independence between the two probabilities makes the algorithm unable to identify the rather complex constellations like parallel octaves and fifths, which would require larger contexts. These parallels occur when two voices move by the same interval (information in $P(I^v_{i-1\to i})$) while the interval between them is a fifth or octave (information in $P(v_i)$). However, the avoidance of such parallels can be formulated as a tangible rule for very specific constellations. Therefore, we think that the best trade-off between n-gram sparsity and the adherence to music theory can be achieved by treating parallel octaves and fifths with special rules discussed in section IV-C.

One final approximation is used, which transforms $P(v_i)$ into a probability that is not dependent on absolute, but only on relative pitch. In order to do this, all notes except for that of the soprano voice are re-encoded as intervals with respect to the next higher voice. In this formulation, the interval probabilities are assumed to be independent from the absolute pitch probability $P(n^S_i)$, which is then removed in order to retain only relative pitch information.

$$\begin{aligned} P(v_i) &= P(n^S_i, I^{A\to S}_i, I^{T\to A}_i, I^{B\to T}_i) \\ &\approx P(I^{A\to S}_i, I^{T\to A}_i, I^{B\to T}_i) \end{aligned} \tag{12}$$

Rather than an approximation, this corresponds to ignoring absolute pitch information, the motivation being that this information would actually hinder the algorithm in following the user's intention. While relative pitch information is important for generating good harmony voicings, absolute pitch information would induce a tendency towards the center of the respective voice ranges. Especially in case of the relatively short frames of a few bars, this tendency could appear unnatural if the user inserts many notes at the borders of voice ranges. Therefore, we think that it is more meaningful to use voice range information only for restricting the search space (see section IV-D).

C. Special Rules

A probabilistic approach to automatic voicing is very suited for capturing relations that are difficult to formulate as tangible rules, or balancing different aspects such as voice movement direction and distance. However, some relations might be easier to capture with another method. The avoidance of parallel octaves and fifths is a good example of such a relation (two voices moving in consecutive octaves or fifths). Their identification requires a relatively large context, which in case of the presented algorithm would conflict with the sparsity reducing approximation in (11). However, this relation can be formulated as a tangible rule, which requires no balancing against other rules, because such parallels are completely prohibited. Therefore, it can be implemented using a penalty factor α_p to suppress the occurrence of parallel motion. Likewise, hidden parallel motion, which occurs when two voices move into an octave or fifth interval while moving into the same direction, is suppressed using a different penalty factor α_hp, which should be larger than α_p. The modified probability is computed as follows.

$$P(v_i|v_{i-1})^* = \alpha_\mathrm{p}^{N_\mathrm{p}(v_{i-1},v_i)}\, \alpha_\mathrm{hp}^{N_\mathrm{hp}(v_{i-1},v_i)}\, P(v_i|v_{i-1}) \tag{13}$$

where $N_\mathrm{p}(v_{i-1}, v_i)$ and $N_\mathrm{hp}(v_{i-1}, v_i)$ denote the number of parallel and hidden parallel motions occurring between v_i and v_{i-1}. The penalty factors α_p and α_hp can be set to 0 in order to completely suppress rule violation, or set to a small number greater than zero in order to allow the user to break the rule without obtaining zero probability solutions.

D. Probability Maximization

The probability maximization method is similar to that of the first step, and utilizes Dijkstra's algorithm to find the optimal voicing sequence. The voicing graph is expanded dynamically and user input accounted for by only allowing paths which contain all notes the user has inserted. Additionally, due to the large amount of possible combinations of four notes, the search space is further restricted by not considering voicings with at least one of the following properties.

- Voice crossings, i.e. a note in a lower voice being higher than that of a higher voice, which is usually avoided.
- Intervals between neighbouring voices that are larger than any interval between those voices observed in the data.
- Incomplete harmonies, i.e. voicings that do not contain all pitch classes contained in the harmony at the given beat. This implies that only one pitch class is duplicated in triad harmonies and none in seventh chords.

While the algorithm will not consider such voicings in the automatic generation of new voicing candidates, the user is still allowed to ignore these constraints. This means that the algorithm can for example handle incomplete harmonies if the user manually inserted them, and while such harmonies might not occur in the data, the probability is non-zero thanks to the application of n-gram smoothing.

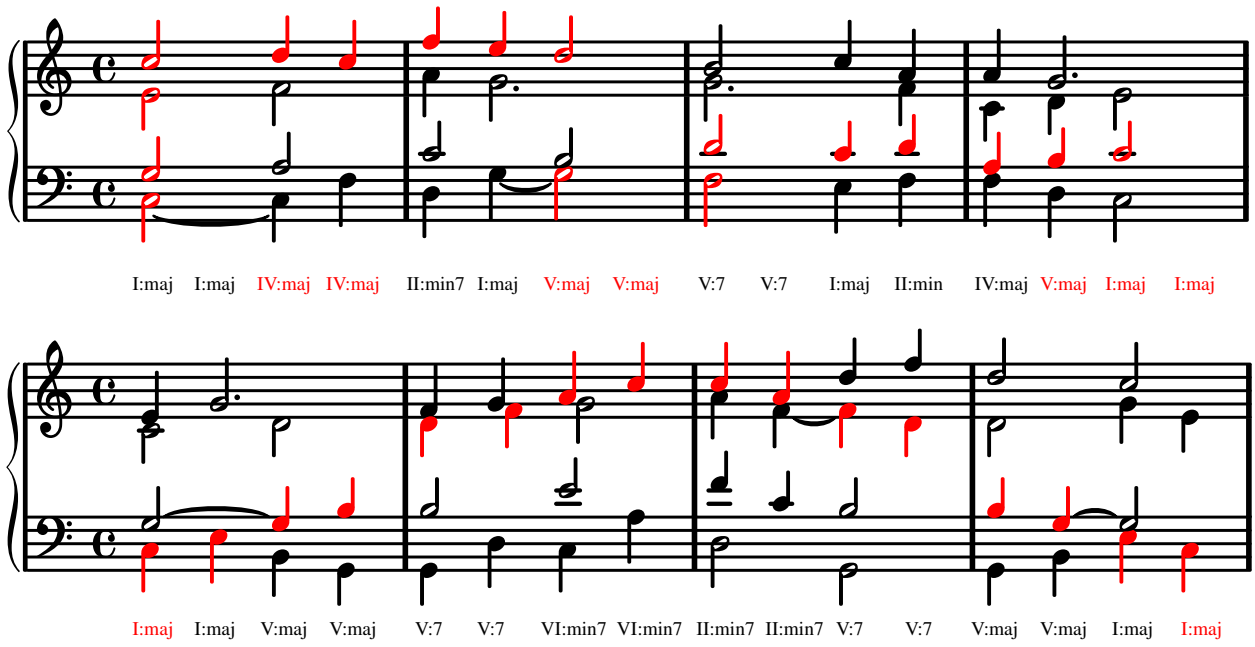

Fig. 3. Two examples of interpolation solutions to constraints that are shown in red. The constraints are inserted by the user and represent his creative intention.

V. Experimental Results

A. Training Data

For the harmony interpolation algorithm, training data has to contain information about both harmonic rhythm and functionality (i.e. information about the key to relate harmonies to). A data set that fulfils both requirements is the KSN annotation data set[16]. After removing sections with uneven signatures (e.g. 3/4), the training data contains about 9100 bars of harmony progressions. To reduce the number of possible n-grams, the chord qualities were restricted to major, minor, diminished, augmented, as well as dominant, major and minor sevenths. More complex qualities were changed to their closest simpler type, e.g. dominant ninths to dominant sevenths.

The training data for voicing interpolation was obtained from the Classical Archives website[17] in the form of MIDI tracks. We chose Johann Sebastion Bach's chorales as training material, which are famous for their four-part voicing. After excluding chorales with more or less than four voices, 380 pieces remain, with a total of about 27000 beats. The voices are read by the program in quarter note resolution, i.e. shorter note ornaments are ignored.

B. Subjective Evaluation

The quality of music is difficult to evaluate objectively due to differences in taste. While in the case of classical chorales, we can evaluate results according to criteria derived from music theory, which is discussed in the next section, we first describe overall subjective impressions that cannot be expressed quantitatively.

Examples of the algorithms output can be seen in Fig. 1, in which the constraints were intentionally set by the user, and in Fig. 8, in which constraints were randomly generated (see section V-C). The melodic rhythms of these examples was generated by automatically connecting repeating notes within a bar, which the voicing algorithm only outputs as quarter notes. To generate harmonically interesting results, the algorithm requires a few constraints in the form of harmonies or partial voicing. Without constraints, the results consist of very common chord progressions like I $\rightarrow$ IV $\rightarrow$ V $\rightarrow$ I, which are musically valid, but can become monotone. However, this problem can be avoided using constraints like partial melodies or uncommon harmonies.

When computed on a single core of a i7-4720HQ CPU, the interpolation of moderately constrained (20% voice notes, 80% empty) pieces that span 4 bars takes on average less than a second. Interpolation of pieces with very many or very few constraints is generally faster. When increasing the length of the interpolation problem to be solved, the required time increases exponentially due to the exponential search space growth. Pieces with 8 bars required on average more than 3 seconds. However, rather than computation time, the inability to generate musical long-range dependencies might limit the meaningful length of interpolation problems: While the generated harmony progressions might be valid, they would, for example, miss repetitive harmonic patterns or motifs, unless manually inserted by the user as constraints. Therefore, there is no real added benefit from interpolating a very long piece, compared to splitting it into smaller pieces to be interpolated. To induce repetition computationally, one

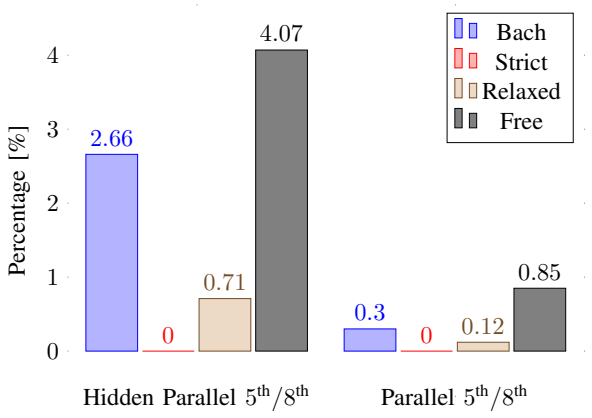

Fig. 4. Occurrence of parallel and hidden parallel motion in Bach's pieces and interpolated results. Note that in case of strict rules, 6 out of 90 interpolation problems did not have a solution, but in case of solvable constraints, parallel motion was completely suppressed.

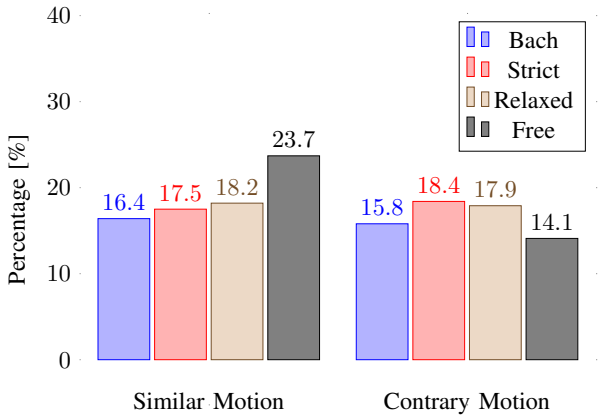

Fig. 5. Ratio of similar (same direction) and contrary motion between voices in Bach's chorales and experimental results. Music theory encourages the use of contrary motion. Since parallel motion is a type of similar motion, the suppression of parallels shifts the balance of motion types.

could possibly expand the model with a context-free grammar for long-range dependencies.

A weakness of the algorithm in its current form is its inability to recognize nonharmonic tones. These are notes that do not belong to the current chord in the underlying harmony progression and are used to embellish the basic voicing. While it would be relatively easy to add such embellishing tones to an automatically generated voicing, the real problem lies in the fact that the algorithm interprets all notes input by the user as harmony constraints. Since the automatic decision whether to treat a note as harmonic tone or embellishment is non-trivial and not yet included in the model, complex note constellations can lead to unnecessary complicated harmony progressions (see Fig. 8) or even unsolvable interpolation problems.

C. Music Theory

Although objective evaluation of music is difficult, in the case of chorales, there are several music theoretical principles a composer should generally follow. While there is no absolute correct in music, and composers sometimes ignore these principles, most rules are very concrete and forbid the usage of specific note constellations. Therefore, we can quantitatively evaluate by counting how often a principle is not followed, and compare the numbers obtained from automatically generated pieces and from pieces composed by Johann Sebastian Bach, who is regarded as a master of chorale composition.

For the experiment, the data set containing 390 chorales was randomly split into 90% training data, from which n-gram statistics were obtained, and 10% test data. From the chorales in the test data set, 100 excerpts spanning 4 bars were randomly extracted. From these excerpts 80% of the contained notes were randomly removed, and the remaining 20% were used as constraints for the interpolation (shown red in Fig. 8). The musical keys of the excerpts were automatically determined by choosing such that the amount of notes not contained in the keys' scales is minimized. Nonetheless, in 10 excerpts the nonharmonic tones were so many that the interpolation algorithm could not find a solution. The experiment was conducted for three different sets of values for the penalty factors α_{h} and α_{hp}, which suppress parallel and hidden parallel motion, respectively.

- Free (no suppression): $\alpha_{\text{h}} = \alpha_{\text{hp}} = 1$, i.e. parallel motion is not explicitly suppressed.
- Relaxed Rules: $\alpha_{\text{h}} = 0.002, \alpha_{\text{hp}} = 0.02$, i.e. parallel motion is suppressed, but allowed when avoiding it is impossible or leads to extremely unlikely results.
- Strict Rules: $\alpha_{\text{h}} = \alpha_{\text{hp}} = 0$, i.e. parallel motion is completely suppressed. In this case, an additional 6 excerpts of the experiment became unsolvable.

While the algorithm computes in quarter note resolution, the chorales of Bach where analyzed in high enough resolution to process every note, e.g. short passing tones which can be used to avoid parallel motion.

The following music theoretical principles were accounted for in the evaluation.

1) Parallel Motion (Fig. 4): Parallel octaves and fifths, i.e. two voices moving from one of said intervals into the same, and hidden parallel octaves and fifths, i.e. two voices moving in the same direction and ending up in one of said intervals, are to be avoided in polyphonic voicing, although the rule for hidden parallels is less strict.

The experimental results confirm the effectiveness of the penalty factors. In case of the strict rule set, 6 additional interpolation problems became unsolvable, because the interpolated chord progressions do not allow a voicing sequence that contains all input notes but no parallel motion. If a problem is solvable, parallel motion is completely suppressed. One could improve the outcome by avoiding parallel motion using nonharmonic tones, or by detecting unavoidable parallels already in the harmony interpolation.

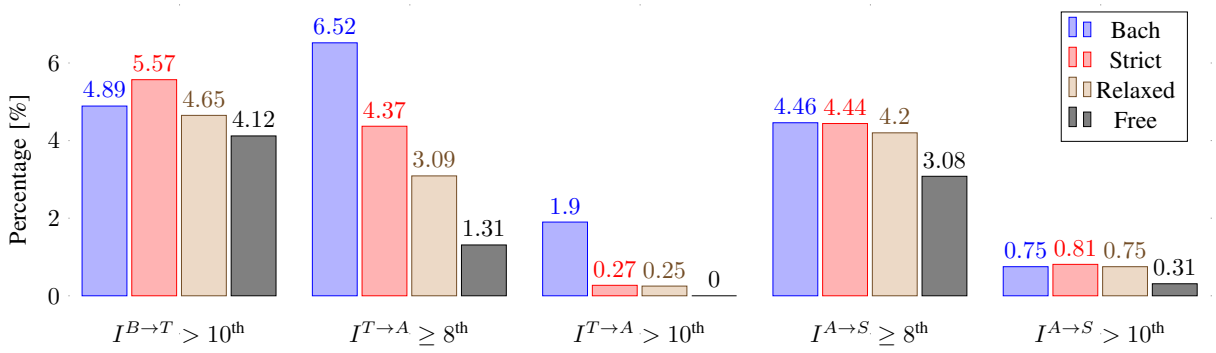

Fig. 6. Occurrence of large intervals between voices in harmony voicings in the experimental data. The superscript indicates the voice pair, e.g. $I^{B\to T}$ is the interval between bass and tenor. Music theory recommends intervals within an octave for the higher three voices, whereas the bass is more free to move, even further from the tenor than a 10^{th} interval, which is usually avoided between higher voices.

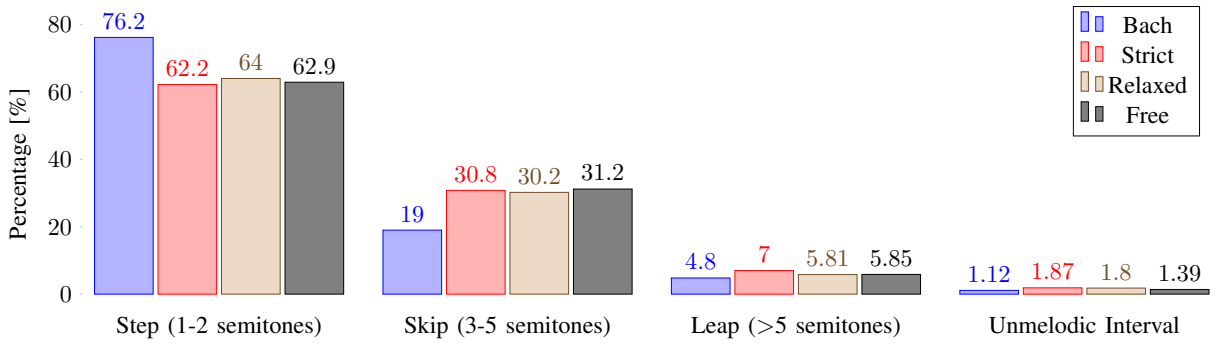

Fig. 7. Properties of melody intervals in the experimental data. Music theory encourages the use of small intervals for smooth voice leading. Large or dissonant intervals are classified as unmelodic (see sqection V-C). One can clearly see the effect of Bach's frequent use of passing tones (often steps between harmonies including nonharmonic tones), which facilitate the composition of smooth melodies and are not yet accounted for in the interpolation algorithm.

2) Relative Motion Direction (Fig. 5): There are three types of relative motions between two voices: Moving into the same direction (similar motion), moving into opposite directions (contrary motion) and only one voice moving while the other one does not (oblique motion). According to music theory, contrary motion is desirable and should occur often.

Interestingly, the evaluation results of the free rule set deviate the most from the properties of Bach's chorales in term of balance between similar and contrary motion. Since parallel motion is a type of similar motion, its suppression shifts the balance towards contrary motion.

3) Inter-Voice Distance (Fig. 6): According to music theory, neighboring voices should stay within a certain distance of each other. As a general rule, they should not be farther apart than a 10^{th} interval. However, this is more important for higher voices, which are encouraged to stay within an octave of each other, whereas the bass usually has more freedom to move away from the neighboring tenor voice.

As can be seen in Fig. 6, the probability maximization approach of the interpolation algorithm tends to keep voices even stricter within these interval boundaries than Bach. Relaxation of the rules allows the algorithm to keep the intervals more often within the boundaries.

4) Smooth Voice-Leading (Fig. 7): In order to obtain smooth melodies, music theory recommends the use of small melody intervals. Furthermore, the following intervals are considered melodic and others should be avoided: Minor and major second, minor and major third, perfect fourth, perfect fifth, ascending minor sixth and perfect octave. For the evaluation result, unison intervals were ignored, since they can also occur in rhythmic function unrelated to voicing.

As can be seen in Fig. 7, the algorithm performs worse than Bach in this respect. Since it does not consider nonharmonic tones, the generation of smooth melodies can be difficult.

5) Final Remarks: The evaluation results indicate that the algorithm is able to quite successfully adhere to music theoretic principles. However, this does not automatically guarantee that the generated results are musically pleasing or interesting. An intuitive example of a musical property not directly covered by music theory is the concept of psychoacoustic consonance. While humans intuitively perceive consonance, it is difficult to measure explicitly. In our model, it is mainly influenced by the voicing structure probability $P(V_i)$ in (11). Since the algorithm does not consider non-harmonic tones, dissonance is no major problem, but subjective evaluation cannot be completely replaced by music theoretic evaluation.

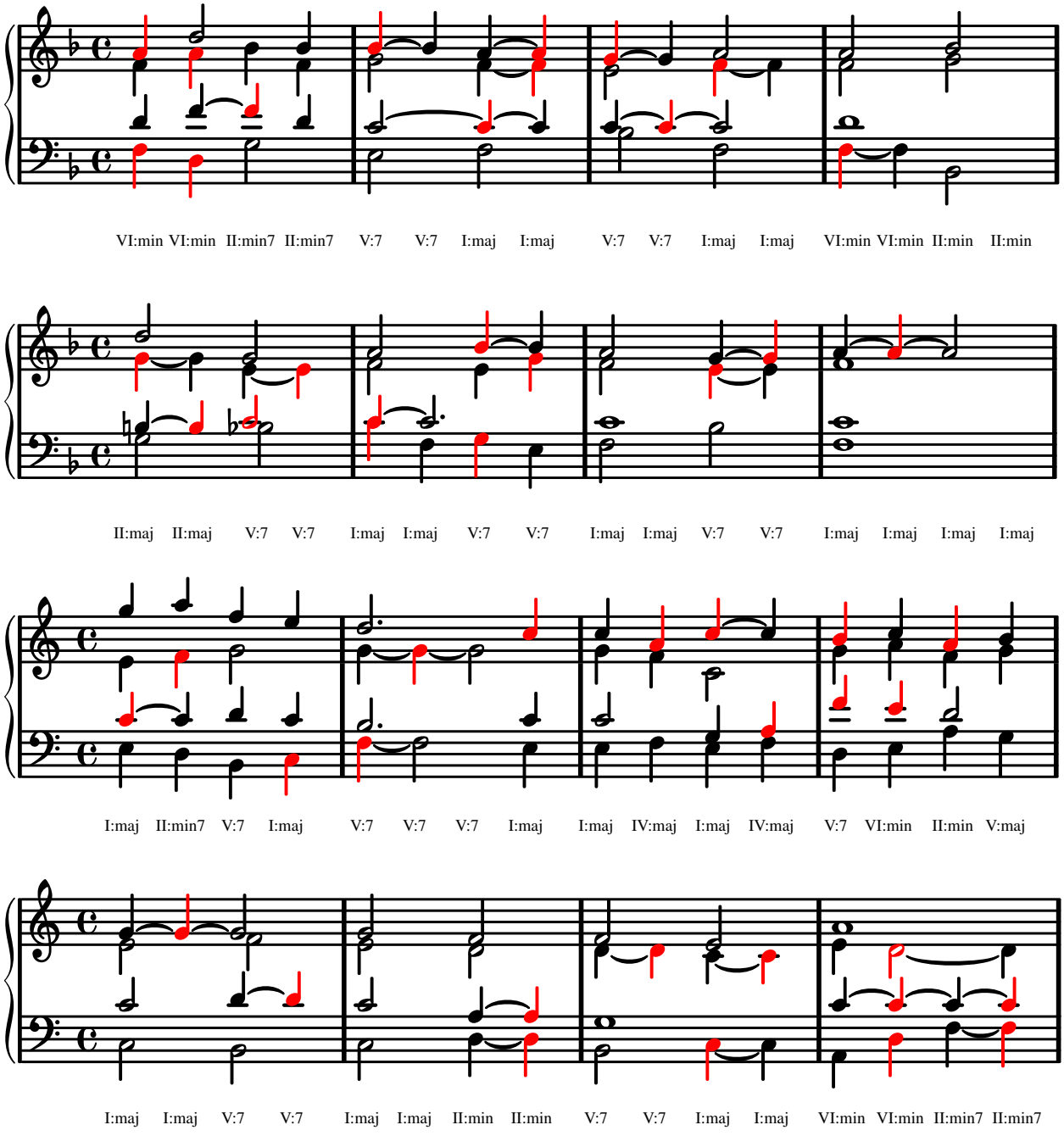

Fig. 8. Four examples of interpolation solutions to constraints that were generated randomly as described in section V-C (shown in red). Since there are no harmonic constraints, the algorithm does not guarantee a harmonic resolution at the end of the interpolated section. Furthermore, due to probability maximization, common harmonies such as I:maj and V:7 are quite dominant in the interpolated results, possibly resulting in less interesting harmony progressions. Also, because the presented algorithm does not account for non-harmonic tones, there are several places in the interpolated pieces, where interpretation of an input note (red) as non-harmonic tone would have allowed to interpolate smoother harmony progressions. Lastly, since the algorithm does not yet consider the possibility of key modulation, the first harmony in the second example is identified as II:maj, whereas interpretation as the secondary dominant V/V would make more sense in the context of music theory.

VI. Conclusions

A. Summary

We made a case for composition assistance that focuses on reacting to the intention of the algorithm's user, based on the motivation to retain the user's sense of accomplishment despite using artificial intelligence. In line with derived design principles, an algorithm was presented that assists a user in composing four-part chorales by first inferring its harmony progression and then computing a suitable voicing. The experimental results show that the algorithm generates valid results when evaluated according to music theoretical criteria. However, since the algorithm does not account for nonharmonic tones, the generation of smooth voice leading is difficult depending on the input constraints.

B. Possible Improvements

Due to the limited context of n-grams, the presented algorithm is not able to recreate long-range harmonic dependencies (e.g. repetitive harmony patterns) found in many types of music. A model for such dependencies, could serve as additional constraint for the interpolation. In case of the voicing interpolation, the context of the voicing n-grams is too small to reliably generate melodic rhythm. This could be addressed by developing a rhythm model to refine the interpolated voicing.

Most importantly, a model of nonharmonic tones would significantly improve the quality of the chorale interpolation. During harmony interpolation, information about how likely a input note is an nonharmonic tone of a certain harmony, could be used to inversely compute how probable the respective harmony is, allowing to more freely handle partial voicing. On the other hand, the same probabilities could be used to embellish the generated voicing with nonharmonic tones, which would significantly facilitate the generation of smooth melodies. One could develop such a model either heuristically based on music theory, or in the ideal case, statistically using data containing both voicing and harmony annotation.

Generally, one could either increase the complexity of the probability model to reproduce more sophisticated properties of the training data, or improve the performance of the algorithm for even faster user interaction. The former could be achieved using a neural network to train probability dependencies given larger contexts, while the latter could be achieved by parallelizing integral parts of the algorithm.

C. Increasing Meaningful User Interaction

The consequent continuation of the research discussed in this paper is to provide the user with more ways to meaningfully influence the music generation process. The drawback of maximizing the harmony progression probability, i.e. the risk of obtaining valid but boring results, could be countered by providing a parameters to tune harmonic tension and/or harmonic complexity throughout the piece to be created. These parameters could shift probability mass from frequent harmonies to more unexpected ones. In the case of voicing, one could think about parameters to tune ornamentation complexity, psychoacoustic consonance or melodic properties.

Such parameters should be abstract enough to allow intuitive use even by inexperienced users. For example, a parameter to directly influence inter-voice distance might be easy to implement, but especially inexperienced users would not likely know how to meaningfully use them, defeating the parameters' purpose of helping realize the user's intention. On the other hand, the parameters should not be too abstract, like for instance emotions, whose complicated connection to music is not yet sufficiently understood. An algorithm that allows to set parameters with meaningful level of abstraction for every beat or bar in the piece would provide a user with powerful tools to create his very own music.

Acknowledgment

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] 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.

[6] Skygge. Hello World. Flow Records. Release: January 12, 2018.

[7] 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.

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

[9] 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.

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

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

[12] 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.

[13] Ricardo Scholz, Emmanuel Vincent, and Frédéric Bimbot. Robust modeling of musical chord sequences using probabilistic n-grams. *ICASSP*, 53–56, 2009.

[14] 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.

[15] 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.

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

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