Polyphonic Voicing Optimization for Automatic Music Completion

Christoph M. Wilk* and Shigeki Sagayama†
* Meiji University, Tokyo, Japan
E-mail: wilk@meiji.ac.ip
† The University of Tokyo, Tokyo, Japan
E-mail: sagayama@74.alumni.u-tokyo.ac.jp

Abstract—**In this paper, we present a new algorithm for automatic music completion. We have proposed automatic music completion as the class of music composition assistance problems of generating a complete piece of music given fragments of musical ideas input by a user. These fragments include partial melodies in multiple voices or parts of the underlying harmony progression. Therefore, it is a generalization of common problems such as melody harmonization or harmony constrained melody generation, but also includes problems with constraints in multiple domains, i.e. multiple voices and harmony. We present a new polyphonic voicing model for automatically completing four-part chorales. It is based on a hidden Markov model and what we call correction factors. These factors are trainable functions that efficiently capture the context of a voicing in order to account for a multitude of music theoretical rules without having to resort to a rule-based system. We observed improvement over our old model with regards to metrics that are derived from music theory for polyphonic voicing, and also invite the reader to try our algorithm themselves at http://160.16.202.131/music_completion_apsipa.**

I. Introduction

Algorithmic music composition is a research topic of increasing popularity. However, many approaches belong to the category of autonomous composition, which means that a human is replaced by a computer, which generates music on its own. On the other hand, our aim is to support human creativity by providing tools for users to more easily realize their own musical ideas. For many of the published automatic music generation systems that do process some sort of input, the type of information that can be input is quite limited. For example, algorithms solving the conventional problem of melody harmonization [1][2] require a single, complete melody as input. However, we want our system to be able to also handle incomplete melodies, in case the user only comes up with a melodic fragment, or to handle notes in multiple voices as well as constraints to the harmony progression, and possibly further modes of input.

We call this problem of processing a variety of user inputs and turning these fragments into a complete music piece “automatic music completion”. As a generalization of several conventional music information tasks such as melody generation, melody harmonization, an ideal solution to this problem is a system that outputs music pieces that reflect the initial ideas of the user, may they be interesting melodic motifs, rhythmic patterns, countermelodies, harmonies, or any combination thereof. In this paper, we apply the principle of automatic music completion to the composition of four-part chorales. This is a popular discipline of classical music and a complex task due to the multitude of music theoretical rules that exists for polyphonic music. We present a new model for polyphonic voicing in combination with an optimization algorithm that overcomes several weaknesses of our previous model [3][4]. In particular, the smoothness of generated melodies was improved and the computation speed considerably increased.

II. Related Research

The first publication on automatic music composition dates back to the 1950s [5]. For an overview over this field of research, we refer to a survey on the topic by Fernández and Vico [6]. Many of the popular approaches to algorithmic composition are of autonomous nature, such as David Cope’s Experiments in Music Intelligence [7], the Melomics music database [8], which was generated using a genetic algorithm, Kulitta [9], a composition framework based on a formal grammar, and Google Magenta’s recurrent neural network which generates human-like piano performances [10]. Despite their popularity, they aim at replacing human composers which does not coincide with our goal.

Music composition systems that support human creativity generally process some sort of user input, which is, however, often quite limited. The composition system Orpheus [11] processes lyrics or an input melody in combination with abstract parameters to generate songs, other support user-driven melody generation constrained by abstract parameters [12], by input melodies in order to generate variations [13], to interpolate missing parts [14], or to improvise a fitting countermelody [15]. The complementary problem of melody harmonization generally involves a melody as user input and generates notes of accompaniment voices [1] or lead sheets [16][17], which are pairs of a single melody and a harmony progressions.

However, our research goal is to not only compute harmonies for melodies or vice versa, but to process any combination of partial harmonies, melodies and voicings. Similar to our approach in that respect is a system called FlowComposer [18], that assist a user in generating lead sheets. The system allows users to constrain both melody and harmony, and generates results using a Markov model. The two approaches

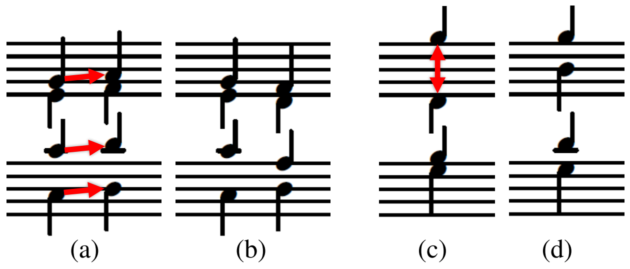

Fig. 1. Two exemplary voicing rules. Parallel fifths and octaves, i.e., the same motion of two voices into a perfect fifth or octave as shown red in (a) should be avoided (as in (b)). Large distances between voices as shown red in (c) should also be avoided (as in (d)).

closest to ours are the four-part chorale generation systems DeepBach [19] based on neural networks, and the similar Coconet [20] based on a convolutional neural network, which both have the same goal, which is to imitate the composer J. S. Bach. While not their focus, both systems can process relatively free input for four voices. However, in contrast to our system, they are based on random sampling, do not allow users to constrain harmony, and are limited to Bach-like music pieces.

III. Automatic Music Completion

A. Free User Input

Our motivation is to provide automatic music composition assistance granting users a lot of freedom to input their own musical ideas. The following two factors can increase this freedom.

1) Allowing input of any size: No input at all, an almost complete melody with a short section missing, or multiple melody fragments.
2) Providing multiple modes of input, e.g., single voice melody, polyphonic voicing, harmony progression, rhythm, or tuning parameters that allow a user to intuitively influence the output.

In this paper, we present a system that provides assistance for composing four-part chorales. Instead of imitating a specific composer, such as J. S. Bach in related research [19][20], our aim is a model that does not extract composer-specific characteristics, but music theoretic principles that are valid for a wider range of music. By extension, this principle could be applied to different musical styles that follow Western music theory.

B. Music Theoretical Rules and Commonness

Musicians have derived a multitude of rules for composing polyphonic music [21], examples of which are given in Fig. 1. However, most of these rules are not absolute, and sometimes ignored by composers. The rules can also conflict with each other, e.g., avoiding parallel fifths and octaves (figure 1a) can result in large intervals between two voices (figure 1c). The problem becomes more complicated when voicing is constrained by user input, which might contain rule violations or prohibit any form of completion without ignoring one or more rules. This is especially important for freedom of user input, which significantly increases when allowing users to break rules. Therefore, we hypothesize that polyphonic voicing can be treated as the optimization problem of violating as few and least important roles as possible. Furthermore, we want to learn the importance of these rules from data.

This requires that patterns that are common in music data conform to music theoretic rules and can thus be learned. However, in this context, it is important to distinguish between commonness and musical necessity. Not every musical pattern that is rare violates a rule of music theory, and if a certain pattern is very common despite its alternatives also adhering to music theory, the probabilistic bias towards this common pattern can actually important probabilistic tendencies to be outweighed. An example is the unison melody interval (repetition of the same note in a melody), which is very common in the training data. In certain melodic contexts, this bias can cause unwanted jumps in melodies, because a large number of unison intervals in combination with one unlikely jump can be overall still more likely than a sequence of smaller intervals without unison intervals encouraged by music theory. This bias towards unison intervals can be further increased by other factors such as preferring certain distance between notes of different voices as our previous model [3][4] did. Therefore, during the development of the presented model, we carefully considered, which information is irrelevant, which rules of music theory should be handled as constraints (section IV-C), and which should be optimized for (sections IV-D to IV-F).

IV. Polyphonic Voicing Model

A. General Approach

To account for the mutual dependence of harmony and voicing, we jointly optimize harmony progression and four-part voicing. To do so, we use a Hidden Markov Model (HMM) as the basis of our model. This is a common approach in the field [22][23] and assumes that listeners expect a certain sequential order of harmonies, which are regarded as hidden states of the observable notes. In our case, these notes are those of soprano, alto, tenor and bass, which in combination comprise a voicing $v_i \equiv (n_S, n_A, n_T, n_B)$. We approach automatic music completion as an optimization problem, and thus formulate the following optimization objective, where H^* denotes an optimal harmony sequence $H \equiv (h_1, \dots, h_N)$ and V^* denotes the corresponding optimal voicing sequence $V \equiv (v_1, \dots, v_N)$.

$$H^*, V^* = \underset{H,V}{\arg\max} \prod_i \left(P(v_i, h_i \,|\, v_{i-1}, h_{i-1}, \dots, v_1, h_1)\right) \tag{1}$$

As a basis, we adopt the following HMM approximation.

$$P(v_i, h_i \,|\, v_{i-1}, h_{i-1}, \dots, v_1, h_1) \approx P(h_i \,|\, h_{i-1})\, P(v_i \,|\, h_i) \tag{2}$$

On top of this HMM basis, we implemented what we call correction factors, which are functions that capture the melodic context of voicings v_i based on music theoretical considerations. These correction factors are trainable from data and are described in sections IV-D to IV-F.

B. Harmony Progression

In this paper, the focus lies on the model of polyphonic voicing and the underlying harmony progression is modeled with relatively simple bigrams. The harmony bigrams $P(h_i \mid h_{i-1})$ capture functional harmony, i.e. the symbols h_i do not represent explicit chords such as a C major triad, but instead functional degrees in the context of a musical key. This information is more relevant for harmony progressions than explicit chords, for example, the first degree chord, e.g. a C major triad in the key of C major, is commonly followed by a fourth degree chord, e.g. a F major triad in the key of C major. However, in the key of G major, the progression from a C major triad to a F major triad would be very unlikely, because F major triads do not even naturally occur in the key of G major.

To increase the flexibility of our model, we include the possibility of key modulation, i.e. the musical keys that the harmonies h_i and h_{i-1} are in can be different. We can train the key modulation probability from data given key annotations and the occurrence of modulations. However, the data available to us that contains such information is a collection of harmony progressions of mainly orchestral pieces, which have significantly different harmonic rhythms than the chorales we aim to model. Therefore, we separate the training of harmonic rhythm and harmonic transition. The harmony bigrams are defined differently depending on whether the harmony changes or not.

$$\begin{aligned} &P(h_i \mid h_{i-1}) \\ &= \begin{cases} P_{\text{same}}(\text{beat of } h_i) & \text{if } h_i = h_{i-1} \\ (1 - P_{\text{same}}(\text{beat of } h_i))\, P^*(h_i \mid h_{i-1}) & \text{else} \end{cases} \end{aligned} \tag{3}$$

where P_{same} denotes the probability that a harmony will continue at the given beat. This probability is assumed to be the same for every harmony and learned from a chorale data set [24]. We expect this probability to be lowest at the first and third beat of a bar, i.e. it is more likely that the harmony changes at the beginning or in the middle of a bar than that it changes at a weaker beat. The harmony transition probability $P^*(h_i \mid h_{i-1})$ is learned from the orchestral data set [25] for all $h_i \neq h_{i-1}$.

C. Basic Voicing

A chord voicing is a concrete realization of a harmony consisting of the four notes $v_i \equiv (n_i^S, n_i^A, n_i^T, n_i^B)$ of soprano, alto, tenor and bass. The voicing probability $P(v_i \mid h_i)$ represents a set of constraints rather than a probability, i.e. the $P(v_i \mid h_i)$ is 1 if the constraints are met and 0 if not. Since this implies that $\sum_{v_i} P(v_i \mid h_i) \neq 1$, it is not an actual probability and we denote it as harmony constraint function $F_H(v_i, h_i)$ in the following.

$$P(v_i \mid h_i) \rightarrow F_H(v_i, h_i) \tag{4}$$

We do not want to normalize $F_H(v_i, h_i)$ in order to obtain a real probability, because this would introduce a bias towards harmonies with more constraints and therefore fewer viable voicings v_i. The constraints are:

- The notes of each voice have to be within the corresponding voice range. Each voice has a range 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, which is typical for choir voices.
- The notes of two neighboring higher voices, i.e. soprano/alto and alto/tenor have to be within the distance of a tenth interval from each other. The bass is allowed to move more freely within the distance of two octaves from the tenor.
- Two nonharmonic tones are allowed per voicing. The algorithm considers passing tones, neighboring tones, escape tones and suspension notes when exploring the search space.
- One chord note can be doubled with the exception of seventh notes. More sophisticated constraints would require information about chord inversion, which is not captured by the current model.

These constraints are relatively weak, leaving a lot of room for user input. In fact, except for the voice ranges, our system can still handle user input that violates these constraints. The listed rules were chosen as binary constraints, because learning them from data often causes unwanted biases and violating them usually does not allow to find better solutions while needlessly increasing the search space. Whether a certain voicing is preferred over another largely depends on how the voices move from the previous voicings. We account for this dependency with the correction factors discussed in the following sections, which increase the probability if a voicing is favorable in a certain context, and decrease the probability if not.

D. Melody Intervals

The first type of melodic context information we consider is how each individual voice moves from one voicing to the next. This information is important, because music theory encourages the use of certain intervals (distance between two notes) for melodies, especially small step movements for smooth melodies. Denoting $x \in \{S, A, T, B\}$, we define the melody interval correction factor F_I as follows.

$$F_I(n_i^x, n_{i-1}^x) = \frac{P(n_i^x \mid n_{i-1}^x)}{P(n_i^x)} \tag{5}$$

If we would not normalize F_I using $P(n_i^x)$, this would introduce a bias towards notes in the middle of a voice range, because very high and low notes are rarer in the data. However, for higher flexibility with regards to user input, we do not want higher or lower notes to be less likely, e.g. if a user intentionally inputs a very high melody fragment, we do not want the algorithm to forcibly move the melody back to the middle of the voice range.

E. Relative Motion

The second type of melodic context information is the relative motion between two voices. This information is important, because music theory forbids the use of parallel fifths and octaves, which describes the state of two voices that are a perfect fifth or octave apart moving by the exact same amount, e.g. the notes (C#, G#) moving to (D, A). Furthermore, voices that are a seventh or ninth apart, should not move by the same interval. For the discussion of relative motion, we introduce the following notation for intervals $I^{x\to y}_{i-1\to i}$.

$$I^{x\to y}_{i-1\to i} \equiv n^y_i - n^x_{i-1} \qquad I^{x\to y}_{i-1\to i} \equiv I^{x\to y}_{i-1\to i} \tag{6}$$

If denoting an interval between two notes of the same voice x or two notes at the same position i, the corresponding arrow is omitted. To capture sufficient information for identifying parallel fifths and octaves as well as consecutive sevenths and ninths, we define the relative motion correction factor F_R as follows.

$$F_R(n^x_i, n^y_i, n^x_{i-1}, n^y_{i-1}) = \frac{P(I^x_{i-1\to i}, I^y_{i-1\to i} \,|\, I^{x\to y}_{i-1})}{P(I^x_{i-1\to i}, I^y_{i-1\to i})} \tag{7}$$

Not normalizing F_R would result in a bias towards small intervals $I^x_{i-1\to i}$ and $I^y_{i-1\to i}$, because they are very frequent in the data. While we do want to encourage the use of such intervals, this is already accounted for by the melodic interval factor F_I (5), and therefore the bias would be too strong.

F. Melodic Motion

The third type of melodic context information is the overall melodic motion of a melody. This information is important, because the consecutive use of large melody intervals, especially in the same direction, should be avoided according to music theory. Important in this context is the concept of steps and skips. The former are the smaller and most common melodic intervals (minor and major second) and the latter are all larger intervals. Based on this, we introduce the following notation for melodic motions $M^x_{i-1\to i}$.

$$M^x_{i-1\to i} = \begin{cases} \text{Skip Up} & \text{if } I^x_{i-1\to i} > 2 \\ \text{Step Up} & \text{if } 0 > I^x_{i-1\to i} \geq 2 \\ \text{Hold} & \text{if } I^x_{i-1\to i} = 0 \\ \text{Step Down} & \text{if } -2 \leq I^x_{i-1\to i} < 0 \\ \text{Skip Down} & \text{if } -2 > I^x_{i-1\to i} \end{cases} \tag{8}$$

To capture consecutive use of intervals, we have to at least consider two previous notes in a melody, leading to the following definition of the melodic motion correction factor F_M.

$$\begin{aligned} &F_M(n^x_i, n^x_{i-1}, n^x_{i-2}) \\ &= \frac{P(M^x_{i-1\to i} \,|\, M^x_{i-2\to i-1})}{P(M^x_{i-1\to i})} \end{aligned} \tag{9}$$

The normalization removes the additional bias towards small intervals, which is already covered by the melodic interval factor F_I (5). Since the definition of $M^x_{i-1\to i}$ entails a significant dimensionality reduction, we can easily capture larger melodic contexts without encountering problems with data sparsity. In our experiments, the melodic motion correction factor F_M considers up to four previous melody notes. An additional benefit resulting from this is that consideration of a larger melodic context reduces the probability of consecutive unisons ($n^x_i = n^x_{i-1}$), i.e. a voice not moving at all. Due to the melodic interval correction factor F_I (5), the probability of unisons is quite high, which is good for efficient voicing, but results in boring results with little motion. Learning larger melodic contexts from data improves the balance between unisons and other melody intervals.

G. Complete Optimization Objective

As described in the previous sections, our approach to voicing optimization consists of first identifying all viable voicings v_i for a harmony h_i using relatively weak constraints, and then weighting these weighting by considering their melodic context. This results in the following voicing optimization objective, which replaces $P(v_i \,|\, h_i)$ in the overall HMM optimization objective (2).

$$\begin{aligned} P(v_i \,|\, h_i) \to \Bigg(&F_H(v_i, h_i) \prod_{x,y\in v \,|\, x>y} F_R(n^x_i, n^y_i, n^x_{i-1}, n^y_{i-1}) \\ &\prod_{x\in v} F_I(n^x_i, n^x_{i-1})\, F_M(n^x_i, n^x_{i-1}, n^x_{i-2}) \Bigg)^{1/w} \end{aligned} \tag{10}$$

where $x, y \in v \mid x > y$ denotes all unique pairs of two voices in a voicing v_i. The weight w serves the purpose of balancing $P(h_i \,|\, h_{i-1})$ and $P(v_i \,|\, h_i)$ in the HMM (2). Since each of the 14 correction terms is derived from a probability, we chose $w = 14$ to balance their influence with the single harmonic probability function. The above voicing optimization objective, although derived from probabilities, is obviously no longer a probability itself, because all correction factors can be larger than 1. However, since it is not possible for loops to exists in the search space (melodies can only move forward), search algorithms are still guaranteed to terminate.

V. Optimization Algorithm

A. Nested Beam Search

The number of possible harmony and note combinations is very large. We denote the range of a voice (number of notes a particular voice can sing) as r and the number of harmony candidates as c and the number of time steps as n. In our model, $r = 25$ due to voice ranges of two octaves and $c \approx 1000$, given the available harmony corpus and considering all possible modulations. In many of our experiments we interpolated pieces of four bars length in eighth tone resolution, resulting in $n = 32$.

We have previously used Dijkstra's algorithm to explore the search space [3], which can have a computational cost of up to $\mathcal{O}((c\,r^4)^n)$. We have improved the computation speed by implementing a beam search algorithm [4], which reduces the complexity to $\mathcal{O}(c\,r^4\,n\,w)$ for beam width w. However, this approach has the drawback that sometimes certain harmonies

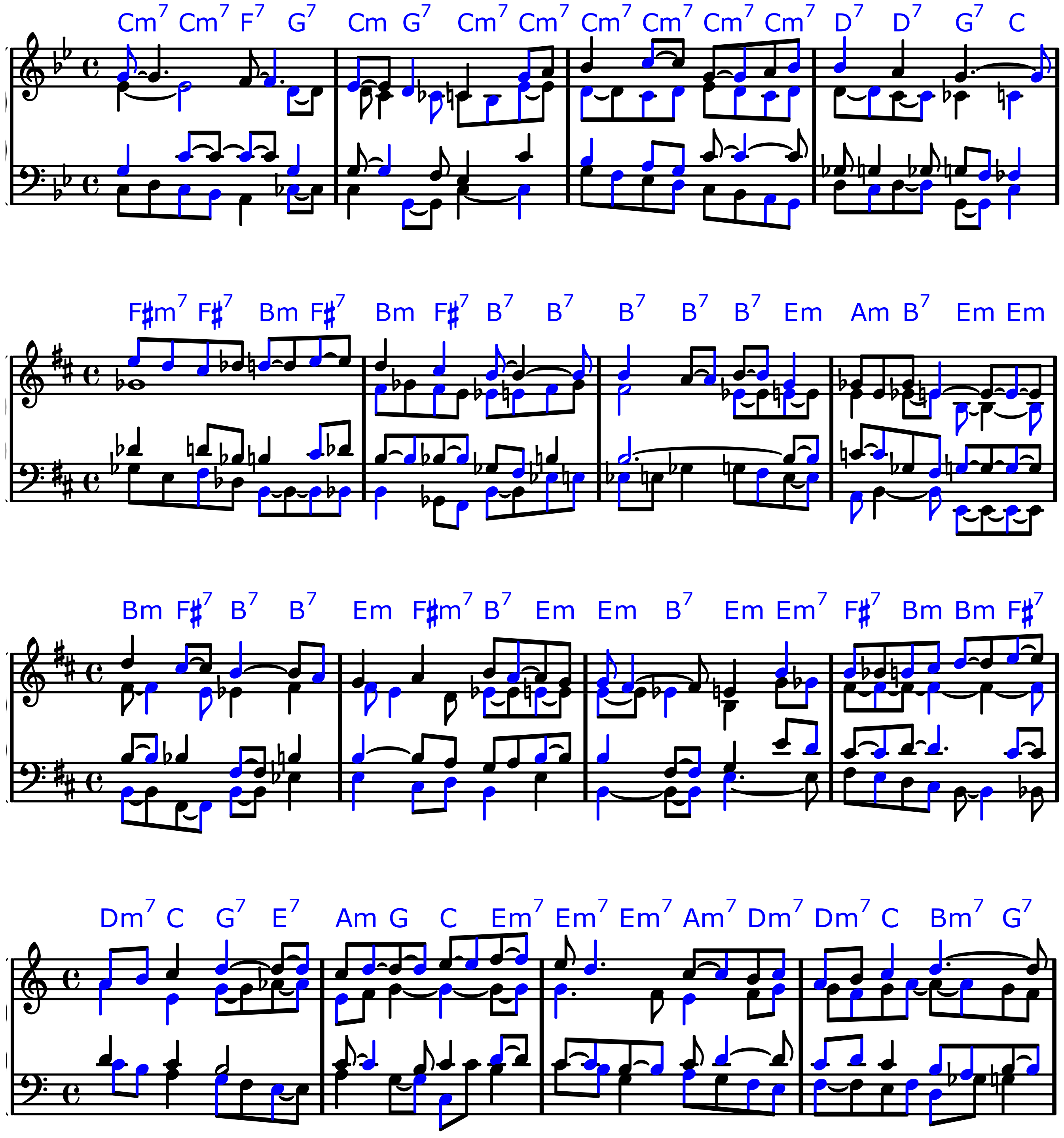
Cm7 Cm7 F7 G7 Cm G7 Cm7 Cm7 Cm7 Cm7 Cm7 Cm7 D7 D7 G7 C
F♯m7 F♯7 Bm F♯7 Bm F♯7 B7 B7 B7 B7 B7 Em Am B7 Em Em
Bm F♯7 B7 B7 Em F♯m7 B7 Em Em B7 Em Em7 F♯7 Bm Bm F♯7
Dm7 C G7 E7 Am G C Em7 Em7 Em7 Am7 Dm7 Dm7 C Bm7 G7

Fig. 2. Four examples of interpolation solutions to constraints (shown in black) that were generated randomly as described in section VI-A. The interpolated harmony progression as well as the generated notes are shown in blue. Since there are no harmonic constraints, the algorithm does not guarantee a harmonic resolution at the end of the interpolated section (as in the third and fourth example).

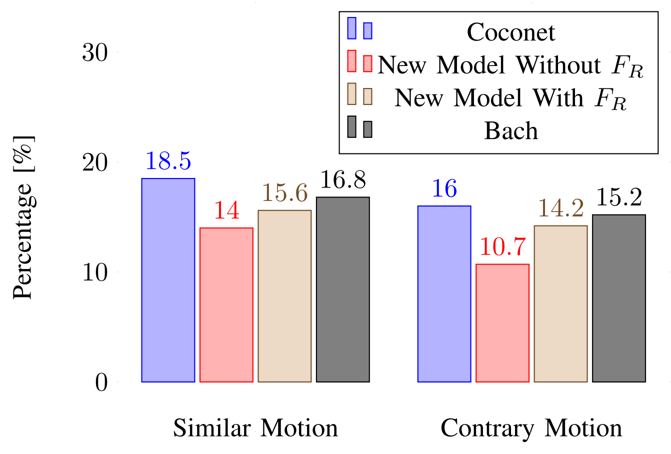

Fig. 3. Ratio of similar (same direction) and contrary motion between voices in Bach's chorales and experimental results. Music theory encourages the use of contrary motion.

become too dominant. For example, in a section without many constraints almost all voicings remaining in the beam can belong to a single harmony. And since the probability for the same harmony to continue is relatively high this dominance becomes increasingly stronger with fewer constraints. If, in this situation, the algorithm encounters a time step with strict user constraints that do not fit well with the dominant harmony, often no smooth harmony transition or voice leading can be found. Therefore, we have implemented a nested beam search algorithm, with a beam width for harmony w_h and for voicing w_v. This algorithm will explore up to w_h harmonies at each time step and up to w_v voicing possibilities for each harmony. In our experiments, we used $w_h = 25$ and $w_v = 40$.

B. Candidate Filtering

For beam search to be effective, the number of dead ends, i.e. harmonies or voicings that have no possible continuation 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 the input notes at this position. It then continues to the previous time step $N - 1$ and removes all harmonies that cannot lead to any harmony remaining for time step N. From the remaining harmonies it again removes all those that conflict with input notes at the corresponding position, and then iteratively continues this procedure until the beginning of the piece is reached.

VI. Evaluation

A. Music Theoretical Metrics

Our goal was to train a model that could generate four-part chorales which adhere to music theory for polyphonic voicing. To evaluate its performance objectively, we then our data set of 271 Bach chorales obtained from the Classical Archives [24] into 90% training data and 10% test data. From the test data we randomly extracted 100 sections of 4 bars length and the randomly removed 50% of the notes in these sections. The remaining notes were used as input constraints for the automatic music completion algorithm. The voicing resolution of both input and output was eighth note resolution and harmonies were interpolated in quarter note resolution.

1) Comparison with Coconet: At the time of writing, the authors were not aware of any other publications with evaluation according to the metrics we propose. A major reason for this is that most approaches to automatic music generation are more concerned with imitating certain styles of music than with music theory itself. To provide a certain degree of comparison, we analyzed 100 randomly chosen Bach-like chorales from the Bach Doodle data set [26], which were generated using Coconet [20]. However, this is not a direct comparison. First of all, the input is quite different. In contrast to our random input constraints, the Bach Doodle data set contains pieces generated from single complete melodies that were input by actual users, and Coconet always generated the same three accompaniment voices. In particular, users might have inserted unmelodic intervals in their input melodies, which would cause worse results for Coconet. Furthermore, the goal of Coconet is to mimic Bach rather than to adhere to music theory. As can be seen in figures 3 through 5, our algorithm significantly outperforms Coconet with respect to several music theoretical metrics.

2) Evaluation Criteria: As evaluation criteria, we implemented several metrics that count how well the rules of music theory of polyphonic voicing were adhered to. Since none of these rules are completely absolute, we also analysed the original 100 sections of Bach's chorales, which were turned into constraints as described above. The goal is to obtain statistics as close as possible to those extracted from Bach's chorales or even more strongly adhere to music theory. Note that since we do not use random sampling, which would naturally result in the same statistics, the achievement of our goal would not be caused by mathematical necessity, but instead confirm our hypothesis that polyphonic voicing can be treated as an optimization problem. In the following, we discuss several aspects of music theory.

3) Relative Motion Direction: 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.

As can be seen in Fig. 3, our algorithm comes quite close to the statistics extracted from Bach's chorales, especially thanks to the relative motion correction factor F_R (7). There is overall a bit more oblique motion in the automatically generated results, indicating that the voices move a bit less than in Bach's chorales.

4) Parallel Motion and Consecutive Intervals: 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

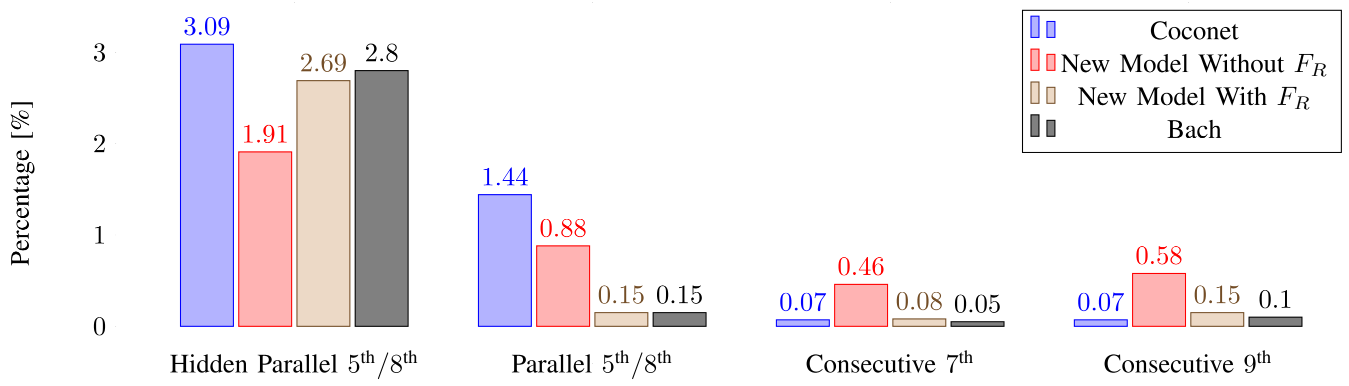

Fig. 4. Occurrence of relative motion that should be avoided. It is successfully suppressed by the learned relative motion correction factor F_R.

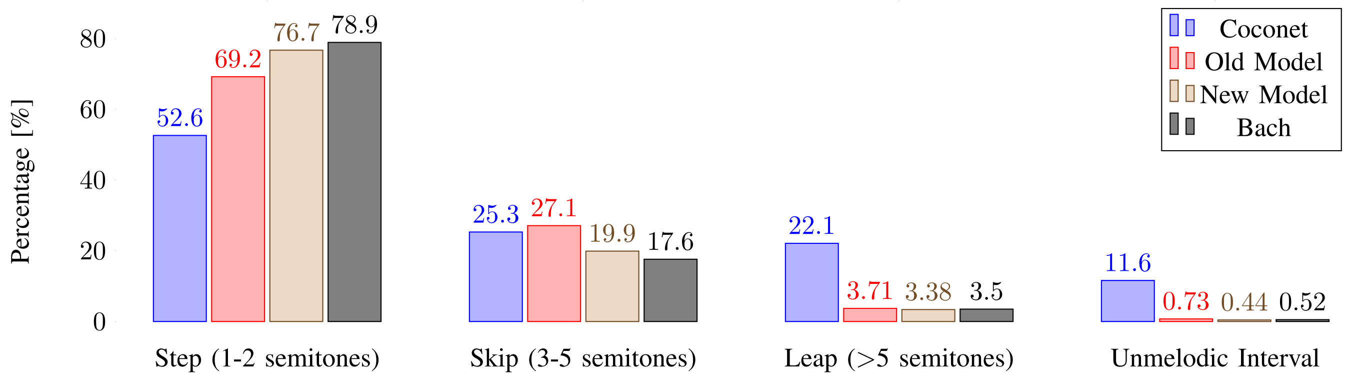

Fig. 5. Occurrence of melody interval types. Music theory encourages the use of small step intervals for smooth voice leading. Large or dissonant intervals are classified as unmelodic (see section VI-A). Our new model generates significantly more steps instead of skips compared to the old model, which leads to smoother voice leading. The results of Coconet might be affected by the fact that users could have input leaps or other unmelodic intervals.

in one of said intervals, are to be avoided in polyphonic voicing, although the rule for hidden parallels is less strict. Also consecutive seventh and ninth intervals between two voices should be strictly avoided.

A weakness of our previous model [3][4] was that the avoidance of parallel motion could not be trained from data, but had to be suppressed by decreasing voicings including such motion with factors requiring manual tuning. In our new model the suppression of parallel motion can be successfully learned from data as can be seen in Fig. 4.

5) Smooth Voice Leading: 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.

Smooth voice leading was a major weakness of our previous model [3][4], where generated results contained around 10% more skip intervals than Bach's chorales. Our new model significantly improves on this point, generating only slightly less step intervals than found in Bach's chorales.

B. *Subjective Evaluation*

Evaluation of music is inherently difficult due to differences in taste. Therefore, we first and foremost invite the reader to try our system themselves at http://160.16.202.131/music_completion_apsipa. In Fig. 2 we provide some exemplary results generated from randomized constraints for music theoretic evaluation as described in the previous section.

The authors impression of the results generated by the presented algorithm is that they are generally pleasing to the ear. Nonharmonic tones are rarely out of place and harmony transitions are generally smooth. When voicing rules are violated it is often not very noticeable or forced by input constraints. The major weakness of the generated results is their lack of structure. The melodies can feel to be moving aimlessly albeit smoothly, and the harmony progressions often lack metrically emphasized resolutions. Therefore, the system still relies on the user to input constraints in order to establish a fundamental structure for harmony and melody. Explicit models for these aspects could further increase the number of tasks that our system can perform without user input.

VII. Conclusion

We presented an algorithm for automatic music completion, which allows a user to freely insert harmonic and melodic fragments for four voices and then generates a complete four-part chorale from those fragments. We discussed a new model for polyphonic voicing, which focuses on contextual information that is relevant for music theory, while avoiding biases that result from common musical patterns that are not necessarily rules. We implemented an efficient nested beam search optimization algorithm that filters harmonies and voicings in a preprocessing step, resulting in significantly improved computation speed, and allowing to easily handle eighth note resolution.

The performance of our algorithm with respect to music theory exceeds our previous results. In particular, the suppression of parallel motion can be learned and does no longer require manual tuning, and the generated melodies are significantly more smooth. To improve our system, we are developing a model for harmony that accounts for key modulation and harmonic resolution, and one could further add a model for melodic phrasing or self-similarity that would increase the structure of melodies.

Acknowledgment

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

References

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

[2] 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.

[3] Christoph M Wilk and Shigeki Sagayama. Harmony and voicing interpolation for automatic music composition assistance. *APSIPA*, 2018.

[4] Christoph Wilk and Shigeki Sagayama. Automatic music completion based on joint optimization of harmony progression and voicing. *submitted to Journal of Information Processing*.

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

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

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

[8] 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.

[9] Donya Quick. *Kulitta: a framework for automated music composition*. Yale University, 2014.

[10] Sageev Oore, Ian Simon, Dieleman Sander, and Doug Eck. Learning to create piano performances. *NIPS Workshop on Machine Learning and Creativity*, 2017.

[11] 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.

[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] Adam Roberts, Jesse Engel, Colin Raffel, Curtis Hawthorne, and Douglas Eck. A hierarchical latent vector model for learning long-term structure in music. *CoRR*, abs/1803.05428, 2018.

[14] Tatsunori Hirai and Shun Sawada. Melody2vec: Distributed representations of melodic phrases based on melody segmentation. *Journal of Information Processing*, 27:278–286, 2019.

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

[16] Alan Freitas and Frederico Guimaraes. Melody harmonization in evolutionary music using multiobjective genetic algorithms. In *Proceedings of the Sound and Music Computing Conference*, 2011.

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

[18] 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.

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

[20] Cheng-Zhi Anna Huang, Tim Cooijmans, Adam Roberts, Aaron Courville, and Douglas Eck. Counterpoint by convolution. *ISMIR*, 2017.

[21] Adolf Bernhard Marx. *Theory and practice of musical composition*. Gordon, 1866. Translated by Saroni, H. S.

[22] Hélene Papadopoulos and Geoffroy Peeters. Large-scale study of chord estimation algorithms based on chroma representation and hmm. In *Content-Based Multimedia Indexing, 2007. CBMI'07. International Workshop on*, pages 53–60, 2007.

[23] Yushi Ueda, Yuki Uchiyama, Takuya Nishimoto, Nobutaka Ono, and Shigeki Sagayama. Hmm-based approach for automatic chord detection using refined acoustic features. In *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pages 5518–5521. IEEE, 2010.

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

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

[26] Cheng-Zhi Anna Huang, Curtis Hawthorne, Adam Roberts, Monica Dinculescu, James Wexler, Leon Hong, and Jacob Howcroft. The Bach Doodle: Approachable music composition with machine learning at scale. In *International Society for Music Information Retrieval (ISMIR)*, 2019.