Gilbert Wassermann and Mark Glickman
Department of Statistics
Harvard University
1 Oxford Street
Cambridge, Massachusetts 02138, USA
gilbert.wassermann@gmail.com,
glickman@fas.harvard.edu

Automated Harmonization of Bass Lines from Bach Chorales: A Hybrid Approach

Abstract: In this article, a combination of two novel approaches to the harmonization of chorales in the style of J. S. Bach is proposed, implemented, and profiled. The first is the use of the bass line, as opposed to the melody, as the primary input into a chorale-harmonization algorithm. The second is a compromise between methods guided by music knowledge and by machine-learning techniques, designed to mimic the way a music student learns. Specifically, our approach involves learning harmonic structure through a hidden Markov model, and determining individual voice lines by optimizing a Boltzmann pseudolikelihood function incorporating musical constraints through a weighted linear combination of constraint indicators. Although previous generative models have focused only on codifying musical rules or on machine learning without any rule specification, by using a combination of musicologically sound constraints with weights estimated from chorales composed by Bach, we were able to produce musical output in a style that closely resembles Bach's chorale harmonizations. A group of test subjects was able to distinguish which chorales were computer generated only 51.3% of the time, a rate not significantly different from guessing.

Ever since the development of music criticism and musicology in the 19th century, the music of J. S. Bach (1685–1750) has held a mythicized position in the writing of music history and the creation of analytical methods. Although creatively groundbreaking, Bach's style is sometimes simplified in music pedagogy to a narrow set of composition guidelines, such as "no parallel fifths" and "contrary motion between bass and soprano." Many of these rules predated Bach and were evident in earlier styles, as evidenced in their formalization by Johann Joseph Fux in his text *Gradus ad Parnassum* (Fux 1971). Musical pedagogy has traditionally taken the view that a student must learn these rules first to fully understand the progression of Western classical music (White 2002). Under this paradigm, students must have a grasp of the most fundamental rules but then must navigate beyond them to learn which of these rules take precedence, thereby building their own ways of understanding the subject. Given the affordability of processing power and the maturity of the field of machine learning, we are now at a stage where we can construct machines that can be trained to learn in a manner similar to a music student. This article proposes a methodology that mirrors this style of learning.

Computer Music Journal, 43:2/3, pp. 142–157, Summer/Fall 2019
doi:10.1162/COMJ_a_00523

Work on automated chorale harmonization fits into two categories: those primarily adopting rule-based systems and those using machine-learning algorithms. Rule-based systems, like those of Kemal Ebcioğlu (1986, 1990), have their roots in traditional Western music theory. For instance, Ebcioğlu created the software Choral, an expert system for which he developed a proprietary logic programming language based on Prolog to incorporate musical rules (Ebcioğlu 1990). In this way, the discipline of music composition is reduced to a constrained optimization problem. This approach requires the user to specify all of the musical constraints, which necessarily relies on the users being knowledgeable about relevant musical rules. Rule-based methodologies often include an exhaustive number of musical rules. For Choral Ebcioğlu included as many as 350 rules, though the implementation did not consider the complex dependency structure among the rules. More recently developed rule-based algorithms, such as those of Tsang and Aitken (1991), incorporated a cohort of 20 rules that is both more lightweight and more malleable. But their framework also did not account for the connections and dependencies among these rules.

In contrast to methods involving the specification of musical constraints, machine-learning systems work by inferring musical patterns (not necessarily explicit rules) given a training corpus of musical data. The state-of-the-art implementation of this

approach is DeepBach (Hadjeres, Pachet, and Nielsen 2017), which used a complex architecture of deep recurrent neural networks to build a large joint probability distribution over all pitches of a chorale across all parts. This probability distribution is then used to perform what Hadjeres, Pachet, and Neilsen refer to as "pseudo-Gibbs sampling," in which proposed pitches are incorporated iteratively into the musical lines until a suitable chorale in the style of Bach is produced. Although this architecture does yield impressive results, the deep-learning algorithm does not permit the deduction of any concrete musical rules or guidelines associated with the replication of what makes the results "sound like Bach." As a black-box algorithm, DeepBach cannot answer this question.

Included under the umbrella of machine-learning approaches are frameworks that model transitions in musical scores as Markov chains. Two publications within this set of algorithms are particularly worthy of mention. The first is the maximum entropy model used by Sakellariou et al. (2016) to generate melodic phrases. Here, the researchers enrich the notion of a Markov chain by superimposing Markov chains of orders 2 through 10 to capture more complex musical dependencies. The second is a strategy implemented by Allan and Williams (2004) that uses two hidden Markov models (HMMs) to perform the tasks of chorale harmonization and ornamentation. One difficulty with approaches that rely exclusively on Markov models is the restriction that the state space focuses only on local musical structure within a chorale. Deep neural networks do not impose such a structural limitation. Given the limited ability of Markov models to go beyond local patterns in Bach's writing, the lion's share of recent publications on automated chorale harmonization have centered on extensions of the neural network framework.

Although previous work is divided between rule-based and machine-learning approaches, the algorithm proposed in this article has its feet firmly in both camps. Its reliance on user-defined constraints means that it is rooted in traditional music theory and, therefore, permits the inclusion of a musician's derived knowledge. The algorithm infers which of these constraints are most important and in what proportion they combine to represent Bach's style. Our approach also allows the incorporation of rule interdependence in the form of interaction terms in our modeling framework. Modeling interactions among rules is essential in reducing the effect of constraint combinations that are especially rare in Bach's style. The estimated weights associated with these "cross constraints" are able to capture the influence of the first-order interactions between the rules, thereby approximating their joint structure.

The architecture of the algorithm proposed in this article is similar to that utilized in Harmonet (Hild, Feulner, and Menzel 1992), in which first the chord structure is generated, then the musical lines. The chord generation process we develop is more closely related to that of Allan and Williams (2004), but takes as its input Bach's bass line as opposed to the soprano line. To our knowledge, our choice of generating chorales from the bass-upwards, as opposed to the melody- or soprano-downwards, has not previously been implemented. Using the melody as a starting point is a natural choice, as the chorales themselves are Bach's harmonizations of contemporary, popular, Lutheran tunes. Music pedagogy has stressed the importance of the bass line, however, as the bass tends to highlight the chords' roots more than the other voices do, and it appears prominently in Baroque practice (exemplified by figured bass notation) as the most important supporting voice for the melody. We have chosen the novel approach of starting with the bass.

One key difference between the approach proposed here and those in previous research is that our algorithm does not seek to include melodic or harmonic embellishments. Some publications, such as those by Hadjeres, Pachet, and Nielsen (2017), build this ornamentation into their methodology by dividing the chorales into smaller rhythmic segments, while others, such as Allan and Williams (2004) propose a special model for ornamentation. The development of a methodology for ornamentation is beyond the scope of this article as we have chosen to focus on Bach's harmonic language. As ornamentation is part of the lifeblood of the Baroque musical idiom, however, we acknowledge the possibility of a connection between composers' harmonic "fingerprints" and the way that they chose to embellish.

The article proceeds as follows. In the following section, we describe our algorithm for generating chorales. The algorithm sequentially infers a harmonic progression through an HMM, and then proposes melody and inner voices of chorales by optimizing a pseudolikelihood based on Boltzmann distributions for choosing the next note. We describe the application of our algorithm in the subsequent section. In addition to implementation details, we develop and report model-based measures of fit, and discuss the results of a survey of test subjects who were tasked with assessing whether a sequence of chorales were written by Bach or were computer-generated by our algorithm.

Methodology

Let $\mathbf{X}$ be a data structure representing all of the information contained within a chorale. We assume that $\mathbf{X}$ contains six subvectors, four of which represent the vocal lines (soprano, alto, tenor, and bass), one vector represents the chords, and the last vector represents the fermata structure. The subvectors representing vocal lines are denoted $\mathbf{x}_S, \mathbf{x}_A, \mathbf{x}_T, \mathbf{x}_B$, with subscripts indicating soprano, alto, tenor, and bass. Because our approach does not attempt to produce embellishments in the style of Bach, each of these subvectors contain n elements, where n is the number of quarter notes in the chorale. For the subvectors representing vocal lines, each of these elements is either a pitch, expressed as a tuple of pitch class and octave, or a rest of quarter note duration. The index, $i = 1, 2, \ldots, n$, represents the number of quarter notes element i is offset from the beginning of the chorale. Thus $\mathbf{x}_S \equiv \{x_{S,1}, \ldots, x_{S,n}\}$ represents the entire soprano line of a chorale. Similarly, the i-th element of the harmonic subvector $\mathbf{C} \equiv \{C_1, \ldots, C_n\}$ contains the set of pitches represented in the generated chord at index i. Each of these pitches is represented by the interval, measured in semitones, that it forms with the tonic modulo 12. The fermata layer is a Boolean vector in which the i-th element is true if the corresponding quarter note has a fermata. This vector is denoted $\mathbf{F} \equiv \{F_1, \ldots, F_n\}$.

The model we propose takes the bass line and fermata structure of the original Bach chorale as the entire input. From this starting point, the architecture of this model can be decomposed into three main parts. First, the input information is used to generate a chordal structure for the chorale, thus predicting $\mathbf{C}$. Next, the bass line and the harmonic structure are used to write the melody ($\mathbf{x}_S$). Given the generated chords and the melody, the inner parts ($\mathbf{x}_A$ and $\mathbf{x}_T$) are filled in. A final "clean up" step is then performed to group together pitches that occur together at the point where a fermata occurs. In this way, the model is able to generate the soprano, alto, and tenor components of $\mathbf{X}$, using only $\mathbf{x}_B$ and $\mathbf{F}$ as input.

Chords

Data for our analyses were obtained through the software package music21 (Cuthbert and Ariza 2010), which contains within its distribution the corpus of all Bach chorales in MusicXML format. Chorales that did not adhere to the unaccompanied SATB (soprano, alto, tenor, bass) format were removed from the corpus. These exclusions resulted in a data set of 354 chorales, 172 of which were determined to be in a major key and 182 of which were determined to be in a minor key. The key of the music was predicted by the Krumhansl-Schmuckler key-finding model (Krumhansl 1990), which compares the appearance of pitches within a piece to ideal distributions for each key signature, returning the key with frequencies of pitch occurrence most similar to those observed in the piece. Our model was developed to propose only pitches within a given key. Chorales with modulations and those classified with an incorrect key signature therefore produced poor results. Chorales in minor key chorales are far more likely to include modulations than major key chorales, so the focus of our analyses involved simulations based on chorales in major keys. From the set of 172 major chorales, 103 were randomly selected to be the training set, leaving 69 chorales as the test set. This proportion of chorales within the test and training sets is similar to that of Allan and Williams (2004).

We model the harmonic structure of a chorale, given its bass line, through HMMs. The chorale's bass line, which is provided as part of the input to the model, is the observed layer of the model, and the chorale's chord progression is the hidden layer. A useful aspect of the bass-up approach is that in Baroque music, bass notes encode more harmonic information than the melody, which should make a bass-up HMM more effective than a melody-down HMM. The dependency structure of the HMM is suited to modeling the bass line as a function of an (unknown) chord sequence as previous chords contain information about future chords, and inner lines are selected from the pitches designated by this harmonic structure. The state space for this model is the set of pitches sounding simultaneously on quarter-note beats in the training data. There were 141 such unique elements for the training set of Bach chorales we selected. The emissions are the twelve semitones in an octave and a rest. A similar approach was presented in Allan and Williams (2004). For our algorithm, the transition probabilities, emission probabilities, and starting state probabilities, all of which are required to fully specify a HMM, are estimated from the training set of chorales. These probabilities are calculated by tallying the number of transitions, emissions, and starting states and dividing by the total number observed in the training set. With these matrices constructed, the most likely chord progression given the bass notes is found by the Viterbi algorithm (Forney 1973).

Because the training set consisted of chorales in different keys, the pitches in these chorales were transformed into their relative notation, that is, the number of semitones they are displaced from the tonic modulo 12. The pitch representations are then arranged in ascending order, and duplicate pitches are removed. This representation eliminates the voicing of the chords, thereby facilitating grouping and comparison by musical function. This relative representation has a one-to-one mapping with the Roman numeral analysis notation used extensively by music theorists (i.e., the pitch classes 0, 4, and 7 are equivalent to the I chord).

One drawback of the HMM approach is that the probability distribution of a chord in this paradigm is conditioned only by the preceding chord. This necessarily omits the possibility of complicated harmonic patterns. Expanding the hidden layer of the state space to include, for instance, the two previous chords rather than the last chord in a higher-order HMM would increase the flexibility of the HMM approach, although a disadvantage is that many two-chord sequences occur too infrequently to result in reliable HMM estimates. Such extensions and others in the context of state-space models for musical composition were considered by Yanchenko and Mukherjee (2017).

Melody and Inner Parts

Our model for the soprano, alto, and tenor parts, given the estimated chordal structure and bass line, proceeds in three steps. First, we identify a set of musical rules that are incorporated into a model for the individual vocal lines. Second, we specify a probability model for an individual element of a vocal line conditional on the other elements and on the proposed musical constraints. These constraints are combined through a linear combination with unknown nonnegative weights. The weights are then estimated by optimizing a regularized pseudolikelihood function based on a set of training chorales. Finally, once the weights are estimated, vocal lines are generated by optimizing a pseudolikelihood only as a function of the weights, the estimated chordal line, and the supplied bass line. We describe this procedure in detail below.

We define $\mathbf{B}$ as a vector of Boolean musicological rules. This vector is further divided into two subvectors $\mathbf{B}_S$ and $\mathbf{B}_C$. The vector $\mathbf{B}_S \equiv \{\mathbf{B}_{S,1}, \ldots, \mathbf{B}_{S,m}\}$ contains m individual rules, each uniquely defined. The single constraints used in this algorithm are given in Tables 1 and 2. The vector $\mathbf{B}_C$ contains indicators of rule pairs, which we term "cross constraints." These cross constraints may be viewed as interaction terms that capture the interdependence between the single constraints. There are at most $\binom{m}{2} = m(m-1)/2$ such cross constraints in $\mathbf{B}_C$. These single constraints and cross constraints take as an argument a musical line defined on all quarter-note indices $i \in \{1, 2, \ldots, n\}$. For notational compactness,

Table 1. Melody Constraints

Constraint	*Behavior Selected for*
Contrary Motion	Soprano line rises as bass line falls or vice-versa.
Follow Direction	Strictly ascending or descending melodic lines.
Stepwise Motion	Melodic interval of a major or minor second.
Leap Then Step	Previous interval greater than major third followed by interval of a major or minor second.
Note to Tonic	If leading tone or supertonic, followed by tonic.
No Parallel Intervals	No consecutive harmonic intervals of perfect fifths or octaves.
No Illegal Jumps	No leaps of a tritone.

Table 2. Inner-Part Constraints

Constraint	*Behavior Selected for*
Stepwise Motion	Melodic interval of a major second or smaller, including unison.
No Parallel Intervals	No consecutive harmonic intervals of perfect fifths or octaves.
No Crossing	Soprano, alto, tenor, and bass pitches in strictly descending order.
New Note	Pitch unheard in other parts.
Octave Max	Intervals from soprano to alto and from alto to tenor that are strictly less than an octave.

we define Θ to be all of the information in the model inputs (bass line, fermatas, and rules) and the generated chords:

$$\Theta \equiv \{\mathbf{x}_B, \mathbf{F}, \mathbf{B}, \mathbf{C}\}. \tag{1}$$

Corresponding to each element in $\mathbf{B}$ is a nonnegative weight, which represents the importance of that constraint to Bach's musical idiom. We let $\mathbf{W}$ denote the collection of all weights. A constraint with a larger weight corresponds to a rule more frequently followed in the training data and therefore more likely to be followed when generating a vocal line.

Rather than specify a true probability model, we adopt a pseudolikelihood approach. This approach involves modeling the probability of an individual element of the vocal line conditionally on the remaining elements in that voice, and using the product of these conditional probabilities as the optimality criterion. This is in contrast to likelihood-based approaches in which the joint distribution of all elements is constructed. The concept of a pseudolikelihood was first proposed by Besag (1974), where it was developed for settings involving graphical models, especially in situations with sufficiently complex temporal and spatial structure. This sequential generation of individual notes is similar to the objective function used in DeepBach (Hadjeres, Pachet, and Nielsen 2017). The pseudolikelihood of the unknown melody weights is given as

$$\mathcal{L}_P(\mathbf{W}|\mathbf{x}_S, \Theta) = \prod_{i=1}^{n} P(\mathbf{x}_{S,i}|\mathbf{x}_{S\setminus i}, \Theta, \mathbf{W}), \tag{2}$$

where $x_{S\setminus i}$ represents the entire original soprano line used by Bach except for the i-th quarter note.

We model the conditional probability mass function of a particular pitch, given the rest of the melodic line, using a Boltzmann distribution:

$$P(\mathbf{x}_{S,i} = k|\mathbf{x}_{S\setminus i}, \Theta, \mathbf{W}) \propto \exp(-f(\mathbf{x}_{S,i} = k, \mathbf{x}_{S\setminus i}|\Theta, \mathbf{W})), \tag{3}$$

where

$$f(\mathbf{x}_{S,i} = k, \mathbf{x}_{S\setminus i}|\Theta, \mathbf{W}) = \sum_{j=1}^{m(m+1)/2} \mathbf{W}_j b_j(\mathbf{x}_{S,i} = k, \mathbf{x}_{S\setminus i}). \tag{4}$$

The sum in Equation 4 is over all individual and cross constraints. Here, $b_j(\mathbf{x}_{S,i} = k, \mathbf{x}_{S\setminus i}) = 1$ if musical line with k as the i-th pitch renders condition $\mathbf{B}_j$ false, and

0 otherwise. Higher values of Equation 4 correspond to poorer imitations of Bach once trained on a corpus of Bach chorales. To prevent overfitting to the training set, we introduce ridge regularization into the pseudolikelihood optimization process to reduce the number of features within the model. We use two different regularization parameters governing the single and cross constraints, respectively, to acknowledge the possibility that the weights for the single and cross constraints might not need to be reduced to the same extent. With the ridge penalties, our optimality criterion can be expressed as

$$\min_{\mathbf{W}\in\mathbb{R}_{\geq 0}^{m(m+1)/2}} \sum_{i=1}^{n} -\log \mathcal{L}_P(\mathbf{W}|\mathbf{x}_S, \Theta) + \lambda_S||\mathbf{W}_S||_2 + \lambda_C||\mathbf{W}_C||_2. \tag{5}$$

Optimizing Equation 5 proceeds in two steps: determining the ridge parameters λ_S and λ_C, then estimating $\mathbf{W}$ given the estimated ridge parameters. The two regularization parameters can be estimated via tenfold cross validation. With these parameters fixed, the weight parameters $\mathbf{W}$ can be found using the Python package CVXPY (Diamond and Boyd 2016), which provides a solver for optimization problems.

The approach to inferring the characteristics of the alto and tenor lines is analogous to the process for inferring the soprano line. A different set of Boolean vectors representing individual and cross constraints are used in the model. These can be found in Table 2. The weights for the corresponding Boltzmann probability model are then estimated using the same procedure as that used for the soprano line.

Having determined the optimal weights, we can now generate musical lines given a bass line and fermata structure. A vocal line is generated by optimizing the following (unregularized) version of the log pseudolikelihood,

$$\min_{\mathbf{x}_S\in\mathbb{R}_{\geq 0}^{n}} \sum_{i=1}^{n} -\log \mathcal{L}_P(\mathbf{x}_S|\Theta, \mathbf{W}). \tag{6}$$

Here, $\mathbf{x}_S$ represents a generated melodic line as opposed to the original melody used by Bach. Also, the optimization is now over a vocal line, where the weights $\mathbf{W}$ have already been estimated in the previous phase of the algorithm.

Equation 6 can be solved by a modified version of simulated annealing proposed by Kitchen and Kuehlmann (2009). This modification was designed to address optimization problems in which the parameters being optimized were binary variables. We extend their approach by recognizing that each binary rule in our model implied by a particular vocal pitch appears as either 0 or the corresponding weight element of $\mathbf{W}$ in the log pseudolikelihood function.

The extended simulated annealing algorithm proceeds in the following manner. The algorithm starts at the first quarter-note index and moves forward in sequence from 1 to n before returning backwards in sequence from n to 1 through the soprano line. At each index, a local search step is performed with probability p_{LS}, a tuning parameter in the algorithm. Otherwise, the simulated annealing process proceeds normally. When p_{LS} is chosen to be close to 0, the algorithm approximates traditional simulated annealing. In a local search step, the algorithm selects the most probable pitch at the given quarter-note index from the set of all possible pitches within the soprano vocal range and in the chord. In this way, we may define the local search step as:

$$\underset{k}{\text{argmax}} \quad P(\mathbf{x}_{S,i} = k|\mathbf{x}_{S\setminus i}, \Theta, \mathbf{W}). \tag{7}$$

If a local search step is not performed, then the usual simulated annealing condition for accepting a proposed pitch applies. Specifically, a proposed pitch leading to a higher pseudolikelihood is accepted, and a suboptimal proposition, k, at quarter-note index i is accepted with probability:

$$P(\text{accept}|k, i) = \exp\left(-\frac{\left(f(\mathbf{x}_{S,i} = k, \mathbf{x}_{S\setminus i}|\Theta, \mathbf{W}) - f(\mathbf{x}_S|\Theta, \mathbf{W})\right)}{T}\right), \tag{8}$$

where $\mathbf{x}_S$ is the generated soprano line and T is the "temperature" parameter at the current iteration of

the simulated annealing process (temperature is a parameter used in simulated annealing to fine-tune the optimization process). This process is continued for 15,000 iterations to produce a generated soprano line that has optimized Equation 6. Over the course of this training process, the negative log–pseudolikelihood function was observed to decrease and converge to a minimum. The algorithm for generating alto and tenor lines is identical, except that each successive iteration switches between proposing an alto note and a tenor note. In addition, generating each successive vocal line is conditioned upon the previous generated vocal lines.

Results and Analysis

In this section we evaluate the chorale harmonizations generated by our algorithm, including the results of a user survey we conducted in which test subjects attempted to distinguish between generated chorale settings and settings composed by Bach.

Chords

We evaluate the model results through the chord sequences derived from the HMM using three approaches. We first examine the comparison of classification rates between the training and test samples. We then examine two related measures based on evaluating the likelihood at the fitted model. We summarize the results here.

To gauge the efficacy of the chord harmonization process we define a metric that we call the *classification rate*. To calculate this metric, we compare the generated chord with Bach's own harmonization at each beat of a chorale. Bach's own harmonization is defined by the output of the music21 automatic chord labeller. We then sum the number of generated chords that match Bach's harmonization and divide by the number of beats in the chorale. A match between Bach's harmonization and the generated chord occurs if the pitch classes of the chromatic scale sounding at the start of a given beat are identical. The histograms of these rates for the chorales in the training and generated test sets are shown in Figure 1. As evidenced in the histograms, the distribution of frequencies of chord occurrences across Bach's chorale settings and our generated versions are similar. A summary of the average classification rates appears in Table 3. We would expect the classification rates for the training chorales to be higher than that of the generated chorales, though the average classification rate of chords in the generated chorales is only slightly lower (62.61% compared to 57.72%). It is worth noting that the comparison of classification rates is a crude approach. The correctness of a classification does not distinguish among different types of chord mismatches. For example, predicting a dominant seventh chord when the original chord is a dominant is counted as a misclassification, even though,these chords serve an identical harmonic purpose.

One of the measures proposed by Allan and Williams (2004) is the average negative log probability of the chord vector given an input. This calculation permits direct comparison between different approaches. Sakellariou et al. (2016) propose a similar approach using the cross entropy of Bach's original harmonization under their model as a measure for evaluating a chorale or melody's generation. Having fit the HMM, the negative log probability of the original Bach chord progression is calculated on the training and test sets. This value is averaged across quarter-note indices to eliminate the effect of chorale length. These values are then averaged over training and test sets, with the results shown in Table 4. Smaller values of this measure indicate better prediction accuracy. The Allan and Williams model achieves a better fit on the training set, but this relationship is reversed on the test set where the Allan and Williams model is outperformed by the bass-up HMM approach. Allan and Williams (2004) point to the sparsity of the matrices used in their HMM as a possible explanation for this overfitting. The same issues are not seen in the bass-up HMM, however, as it has been trained on even fewer chorales (103 as opposed to 121).

Another likelihood-based measure we consider is point entropy. This is used by Conklin and Witten (1995) to investigate which aspects of a chorale melody are the least surprising given their model. In an entropy profile, lower values correspond to

Figure 1. Classification rates for chords in training and test sets.

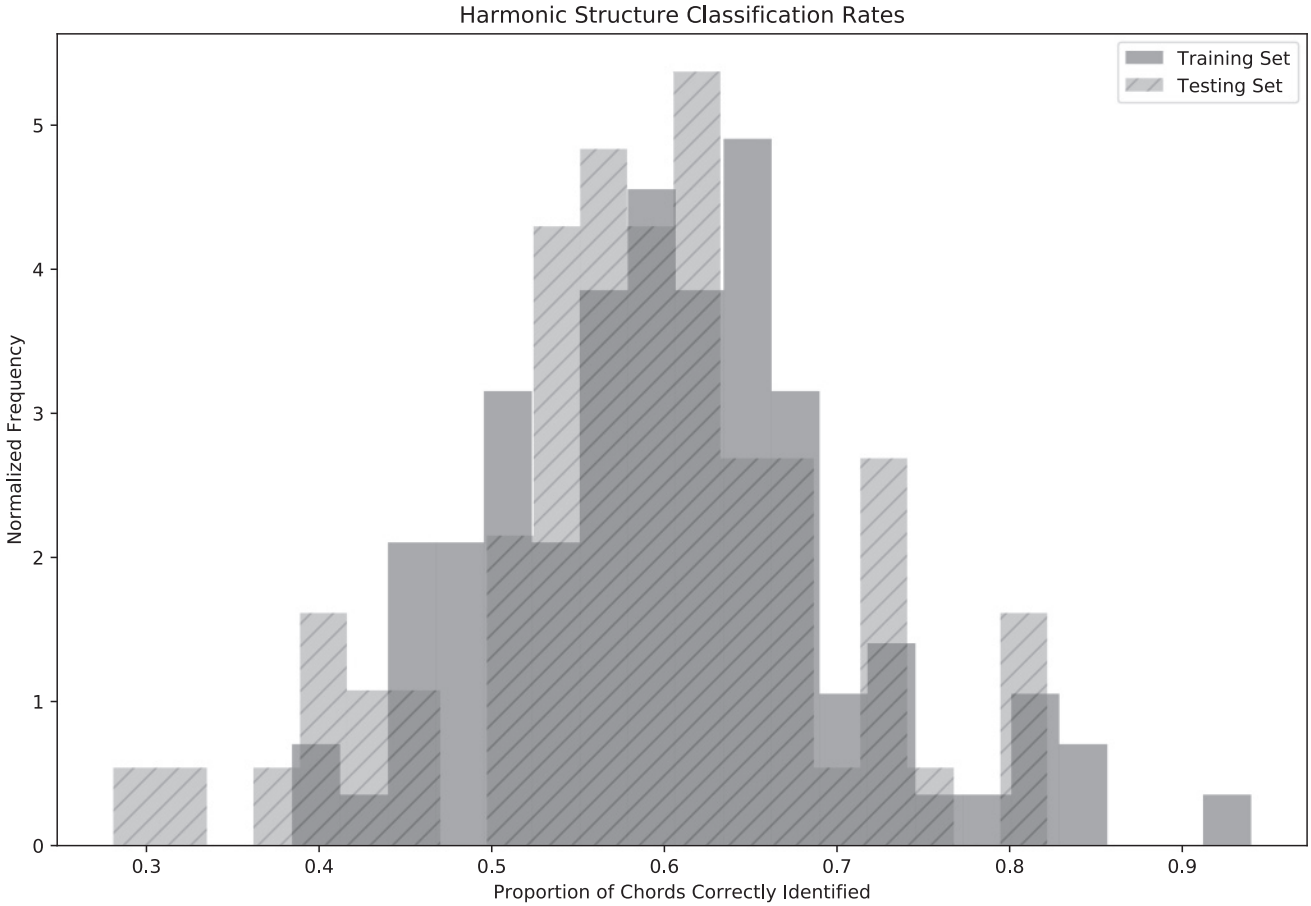


Table 3. Mean Classification Rates for Bass-Up HMM

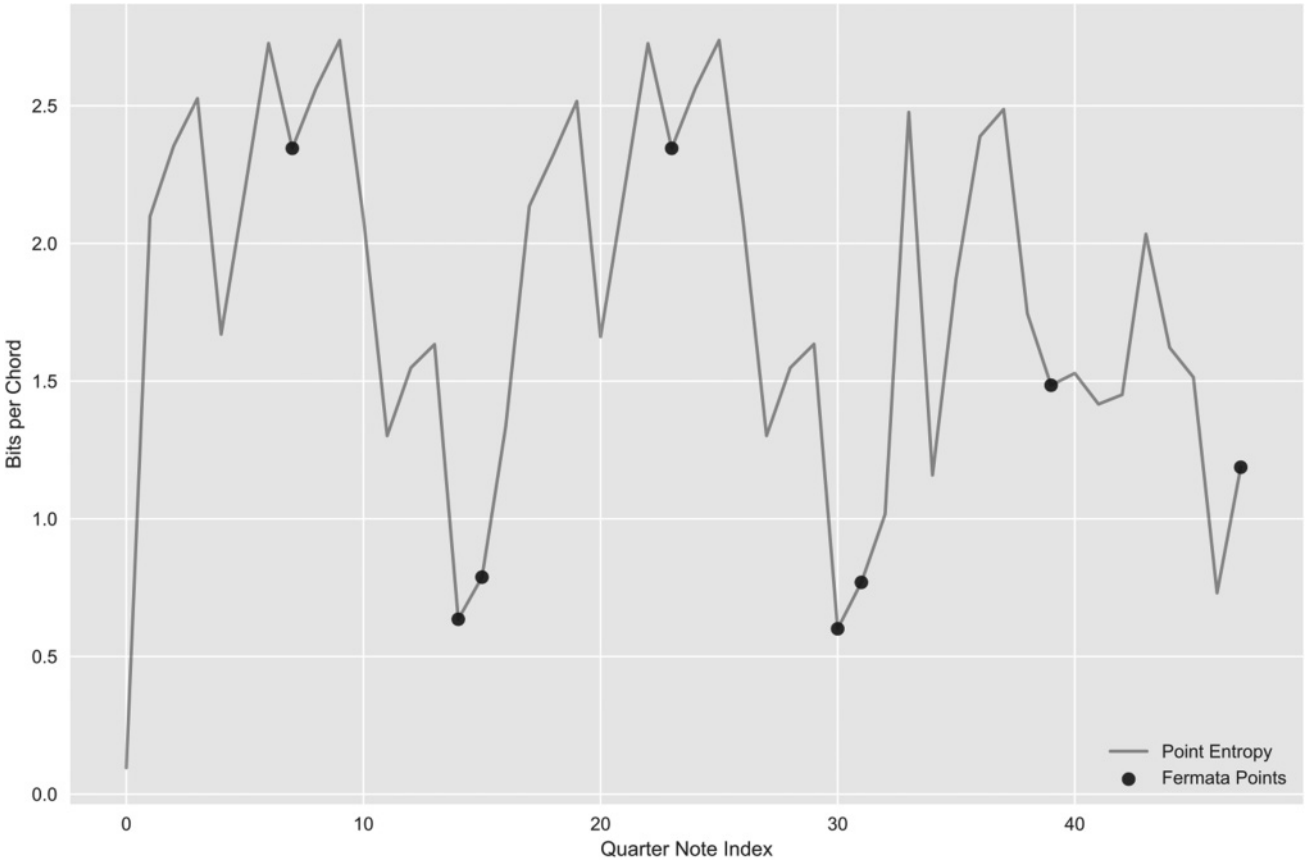
Parts of Chorale	*Bach Chorales*	*Generated Chorales*
Full Chorale	62.61%	57.72%
Cadential Points	86.72%	79.19%

Table 4. Average Negative Log Probability of Chord Vector

Method	*Training Chorales*	*Test Chorales*
Bass-up HMM	4.16	4.31
Allan and Williams (2005)	2.56	4.90

chords more likely to be selected by Bach. These entropy profiles allow exploration of the model's behavior at predicting the structure of cadential points. Consider, for example, the chorale BWV 248(5)/53 in the test set (verse 9 of "Ihr Gestirn, ihr hohlen Lüfte" from the Christmas Oratorio, Part V), whose entropy profile is shown in Figure 2. Chords with fermatas are marked with dots. As shown in the figure, the fermatas are almost always found at local minima. This parallels the musicological approach to harmonization, in which cadential points are almost always the most deterministic

Figure 2. Entropy profile for a chorale in the test sets. The chorale used here is verse 9 of "Ihr Gestirn, ihr hohlen Lüfte" from part V or the Christmas Oratorio, BWV 248(5)/53.



parts of a chord progression. It also signifies that the bass-up HMM predicts "plausible cadences," putting it on par with Allan and Williams' (2004) approach. This plausibility is further confirmed as the distribution of classification rates at cadential points is shown to be far higher than the rates over the full chorale.

The average classification rates at cadential points are shown in Table 3 in which the test chorales have a high classification rate (about 79%), not much lower than that of the training set (about 87%). Based on these analyses, we can conclude that the bass-up HMM approach is a strong generative model for the harmonic structure of Bach chorales. The bass-up HMM does not suffer from the same degree of overfitting as the best melody-down HMM (Allan and Williams 2004), while maintaining the same ability to more accurately predict the chords at cadential points.

Melody and Inner Parts

The estimated weights for the melody are given in Table 5. The single constraints have a stronger effect on the chorale harmonization than the cross constraints, which is reflected through the higher weights. The lower weights of the cross constraints is partly a result of the higher estimated regularization penalty for cross constraints. Of the single constraints, those governing the shape of the melody line have the largest weights (Contrary Motion, Stepwise Motion, Follow Direction), whereas those that are predicated on more codified music

Table 5. Learned Weights of Melody Constraints

Single Constraint	*Weight*	*Cross Constraint*	*Cross Weight*
No Illegal Jumps	0.000008	No Parallel Intervals	0.000002
		Contrary Motion	0.000002
		Note to Tonic	0.000002
		Leap Then Step	0.000002
		Stepwise Motion	0.000002
		Follow Direction	0.000002
No Parallel Intervals	0.100000	Contrary Motion	0.005000
		Note to Tonic	0.000002
		Leap Then Step	0.005000
		Stepwise Motion	0.005000
		Follow Direction	0.010000
Contrary Motion	0.350000	Note To Tonic	0.005000
		Leap Then Step	0.015000
		Stepwise Motion	0.025000
		Follow Direction	0.025000
Note To Tonic	0.150000	Leap Then Step	0.005000
		Stepwise Motion	0.010000
		Follow Direction	0.010000
Leap Then Step	0.300000	Stepwise Motion	0.020000
		Follow Direction	0.020000
Stepwise Motion	0.450000	Follow Direction	0.025000
Follow Direction	0.400000	–	–

theory concepts (No Illegal Jumps, No Parallel Intervals) have the lowest weights. This counterintuitive result could be explained by the fact that patterns that are "forbidden" never appear in the training set.

The opposite is seen from the estimated weights for the inner parts, shown in Table 6. Here, the pure music-theoretical constraints (New Note, No Crossing) have the largest weights, whereas stylistic constraints (Stepwise Motion) are of secondary importance. Once again, the cross constraints are substantially smaller in part owing to the more aggressive regularization parameter. A possible explanation for the difference in stylistic and structural preferences between melody parts and inner parts could be that the the melody is required to be "musical" (easier to sing and remember), whereas the inner parts often serve a more harmonic function.

Full Chorales

Although this algorithm has the potential to produce chorales that match Bach's musical idiom closely, it does not reach this high level consistently. Some chorales that include modulation or are assumed to be in the wrong key can produce jarring dissonances. However, there are some chorales that stand out as musically intriguing and similar in style to Bach. Whether these particularly attractive chorales are comparable to or surpass the level of those generated by Hild, Feulner, and Menzel (1992) is subject to debate. If this method can be considered comparable

Table 6. Learned Weights of Inner-Part Constraints

Single Constraint	*Weight*	*Cross Constraint*	*Cross Weight*
No Crossing	0.02050	Stepwise Motion	0.00110
		No Parallel Intervals	0.00095
		Octave Max	0.00065
		New Note	0.00140
Stepwise Motion	0.01650	No Parallel Intervals	0.00070
		Octave Max	0.00040
		New Note	0.00105
No Parallel Intervals	0.01450	Octave Max	0.00030
		New Note	0.00100
Octave Max	0.00750	New Note	0.00035
New Note	0.02050	–	–

Table 7. Chorales in User Survey

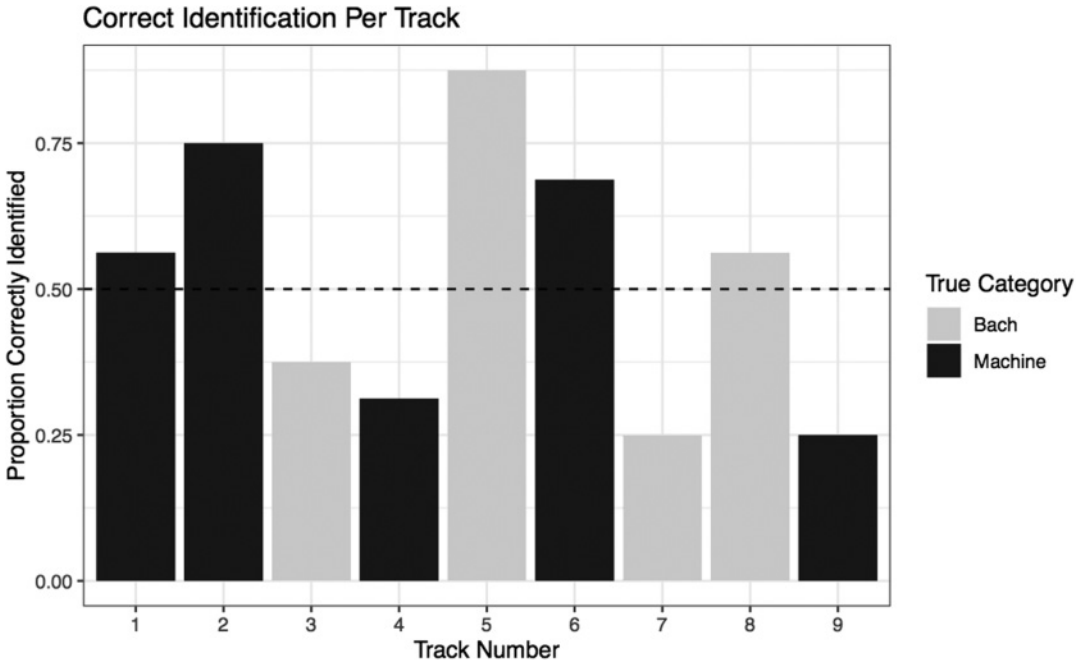
BWV	*Title*	*Setting*
398	O Gott, du frommer Gott	Generated (Track 1)
407	O wir armen Sünder	Generated (Track 2)
55/5	Werde munter, mein Gemüte, verse 6; from the Cantata *Ich armer Mensch,* ich Sündenknecht	Generated (Track 3)
294	Der Tag, der ist so freudenreich	Generated (Track 4)
375	Lobt Gott, ihr Christen, allzugleich	Generated (Track 5)
248/12	Ermuntre dich, mein schwacher Geist, verse 9; from the Christmas Oratorio, Part II	Bach
303	Ein feste Burg ist unser Gott	Bach
346	Ich dank dir, Gott, für all Wohltat	Bach
348	Ich dank dir, lieber Herre	Bach

to Harmonet, however, we may conclude that the musical quality is consistent with that of an improvising organist. At the very least, these generated chorales challenge the assertion set forth by Fernández and Vico (2013) that constraint-based methodologies always produce "aesthetically displeasing" chorales, since we can assume that an unsatisfactory chorale differs from Bach's style.

We conducted a survey to assess whether musicians could distinguish between bass-up HMM generated chorales and those by Bach. Previous work, including Huang et al. (2017), evaluated performance through surveying respondents who listen to generated music. We recruited 16 music students who were asked to listen to nine pieces of music: five of which were generated by the algorithm and four of which were Bach's setting (see list of chorales used in Table 7). Recordings of the generated chorale settings can be found at www.mitpressjournals.org/doi/suppl/10.1162/COMJ_a_00523.

The choice of generated chorales were among those without jarring dissonances, so we note that they were not a random selection. Nine of these subjects were students studying music at Oxford University, and the remaining seven were members of the Harvard-Radcliffe Orchestra. Our survey included only music students because they

Figure 3. Proportion of correct responses to each track in poll.



have a predetermined idea of the sound of a Bach chorale. For this group, listening to the tracks did not substantially alter their knowledge of the subject. By contrast, a listener unfamiliar with Bach chorales might formulate their own spurious patterns over the course of the test, which would alter their responses. Hadjeres, Pachet, and Nielsen (2017) extend this approach by further partitioning the polled group on their (self-reported) musical knowledge. The melody and inner voices of the original Bach chorales were quantized to the quarter note to remove ornamentation, so that they were comparable to the generated chorales. These test subjects were then asked to select whether the chorale they had just heard was produced by a machine or composed by Bach. The distribution of correct responses is shown in Figure 3. Tracks 2 and 5 had the greatest proportion of correct guesses, but the overall pattern appears to suggest a lack of ability to discriminate chorales generated from our algorithm from chorales composed by Bach. Two example outputs are included in the Appendix.

We analyzed the results to examine whether the subjects, on average, were performing significantly better than guessing. Among the tracks that were generated by our algorithm, only 51.3% were guessed correctly. Similarly, among the tracks that were written by Bach (but quantized to remove ornamentation), only 51.6% were guessed correctly. Assuming independence among responses to tracks and across survey respondents, the p-value for the significance of having a better than 0.5 probability of guessing correctly in each case was greater than 0.25, a highly nonsignificant result. These results suggest that the respondents performed, on average, no better than guessing at random, and suggest that the quality of the generated chorales was indistinguishable from that of the original, but quantized, harmonizations.

Conclusion

We have developed an algorithm for generating Bach chorales that explored two new approaches to the problem. First, chorales were generated from the bass up, as opposed to the melody down. This appears to have an improvement in out-of-sample performance

over implementations that take the melody of a chorale as an input. Second, the generating model combined musicology and machine-learning methodologies, each of which have been explored separately in previous work. This framework was tested on a group of test subjects who were found unable to reliably distinguish between the generated chorales and those written by Bach.

The choice of seven single constraints for the soprano and five constraints for the inner parts stands in stark contrast to previous approaches, which included constraints numbering in the tens or hundreds. Our model permits a greater number of constraints, involving both marginal and higher-order interactions, and the degree to which constraints are relevant is a feature of our modeling approach. Our current model with only twelve rule-based constraints is shown to achieve a good fit to the test set, however, and so it is not clear that a more complex model would perform better.

This article has assessed the efficacy of the generative model by comparing chorales composed by Bach to those generated using only Bach's original bass line and fermata structure. This trained model could also be used to generate chorales from user-defined inputs, however. In this way, bass lines and fermata structures not composed by Bach could be used to generate a chorale in his style, without using any of Bach's own material. The ability of the model to accept inputs not taken from Bach's compositions is also a feature of DeepBach (Hadjeres, Pachet, and Nielsen 2017).

Future work using this framework may seek to compare compositional styles. This model produces a model-dependent representation of a composer's unique style as a weighted set of constraints. Comparing the representations of different composers might provide further insight into the development of Western classical music. An example of such research might compare the model presented in this article with a model trained on Telemann's chorales.

The framework presented here, which uses fairly standard modeling components and incorporates Western classical rules of harmony in a "soft" manner is shown to produce results that are comparable to those of cutting-edge machine-learning approaches. Although ongoing work on algorithmic music generation appears increasingly weighted towards these deep neural network architectures, the strong performance of this algorithm should encourage researchers to reconsider and reincorporate the field's musicological roots.

References

Allan, M., and C. K. I. Williams. 2004. "Harmonising Chorales by Probabilistic Inference." In *Advances in Neural Information Processing Systems Conference Proceedings*, pp. 25–32.

Besag, J. 1974. "Spatial Interaction and the Statistical Analysis of Lattice Systems." *Journal of the Royal Statistical Society: Series B (Methodological)* 36(2):192–236.

Conklin, D., and I. H. Witten. 1995. "Multiple Viewpoint Systems for Music Prediction." *Journal of New Music Research* 24(1):51–73.

Cuthbert, M. S., and C. Ariza. 2010. "Music21: A Toolkit for Computer-Aided Musicology and Symbolic Music Data." In *Proceedings of the International Conference on Music Information Retrieval*, pp. 637–642.

Diamond, S., and S. Boyd. 2016. "CVXPY: A Python-Embedded Modeling Language for Convex Optimization." *Journal of Machine Learning Research* 17(1):2909–2913.

Ebcioğlu, K. 1986. "An Expert System for Chorale Harmonization." In *Proceedings of the AAAI National Conference on Artificial Intelligence*, pp. 784–788.

Ebcioğlu, K. 1990. "An Expert System for Harmonizing Chorales in the Style of J. S. Bach." *The Journal of Logic Programming* 8(1):145–185. Special Issue: Logic Programming Applications.

Fernández, J. D., and F. Vico. 2013. "AI Methods in Algorithmic Composition: A Comprehensive Survey." *Journal of Artificial Intelligence Research* 48(1):513–582.

Forney, G. D. 1973. "The Viterbi Algorithm." *Proceedings of the IEEE* 61(3):268–278.

Fux, J. J. 1971. *The Study of Counterpoint: From Johann Joseph Fux's* Gradus ad parnassum, trans. A. Mann. New York: Norton.

Hadjeres, G., F. Pachet, and F. Nielsen. 2017. "DeepBach: A Steerable Model for Bach Chorales Generation." In *Proceedings of the International Conference on Machine Learning*, pp. 1362–1371.

Hild, H., J. Feulner, and W. Menzel. 1992. "HARMONET: A Neural Net for Harmonizing Chorales in the Style of J. S. Bach." In J. E. Moody, S. J. Hanson, and R. P. Lippmann, eds. *Advances in Neural Information Processing Systems 4*. Burlington, Massachusetts: Morgan Kaufmann, pp. 267–274.
Huang, C. A., et al. 2017. "Counterpoint by Convolution." In *Proceedings of the International Conference on Music Information Retrieval Conference*, pp. 211–218.
Kitchen, N., and A. Kuehlmann. 2009. "A Markov Chain Monte Carlo Sampler for Mixed Boolean/Integer Constraints." In A. Bouajjani and O. Maler, eds. *Computer Aided Verification* volume 5643. Berlin: Springer, pp. 446–461.
Krumhansl, C. L. 1990. *Cognitive Foundations of Musical Pitch*. Oxford: Oxford University Press.
Sakellariou, J., et al. 2016. "Maximum Entropy Models Capture Melodic Styles." *Scientific Reports* 7(9172):Art. 9172. Available online at www.nature.com/articles/s41598-017-08028-4. Accessed February 2020.
Tsang, C. P., and M. Aitken. 1991. "Harmonizing Music as a Discipline in Constraint Logic Programming." In *Proceedings of the International Computer Music Conference*, pp. 61–64.
White, J. D. 2002. *Guidelines for College Teaching of Music Theory*, 2nd ed. Lanham, Maryland: Scarecrow.
Yanchenko, A. K., and S. Mukherjee. 2017. "Classical Music Composition Using State Space Models." Available online at arxiv.org/pdf/1708.03822.pdf. Accessed February 2020.

Appendix: Example Outputs

Two of the harmonizations generated by our algorithm from Bach bass lines are shown in the following. Recordings of these harmonizations are at at www.mitpressjournals.org/doi/suppl/10.1162/COMJ_a_00523. The first is a harmonization of the chorale "Werde munter, mein Gemüte," from the cantata *Mensch, ich Sündenknecht* BWV 55 (Figure 4, cf. Sound Example 3). The second is a harmonization of the chorale "O Gott, du frommer Gott," BWV 398 (Figure 5, cf. Sound Example 1).

Figure 4. The automated harmonization of the bass line from Bach's chorale "Werde munter, mein Gemüte," BWV 55/5. Sound Example 3.

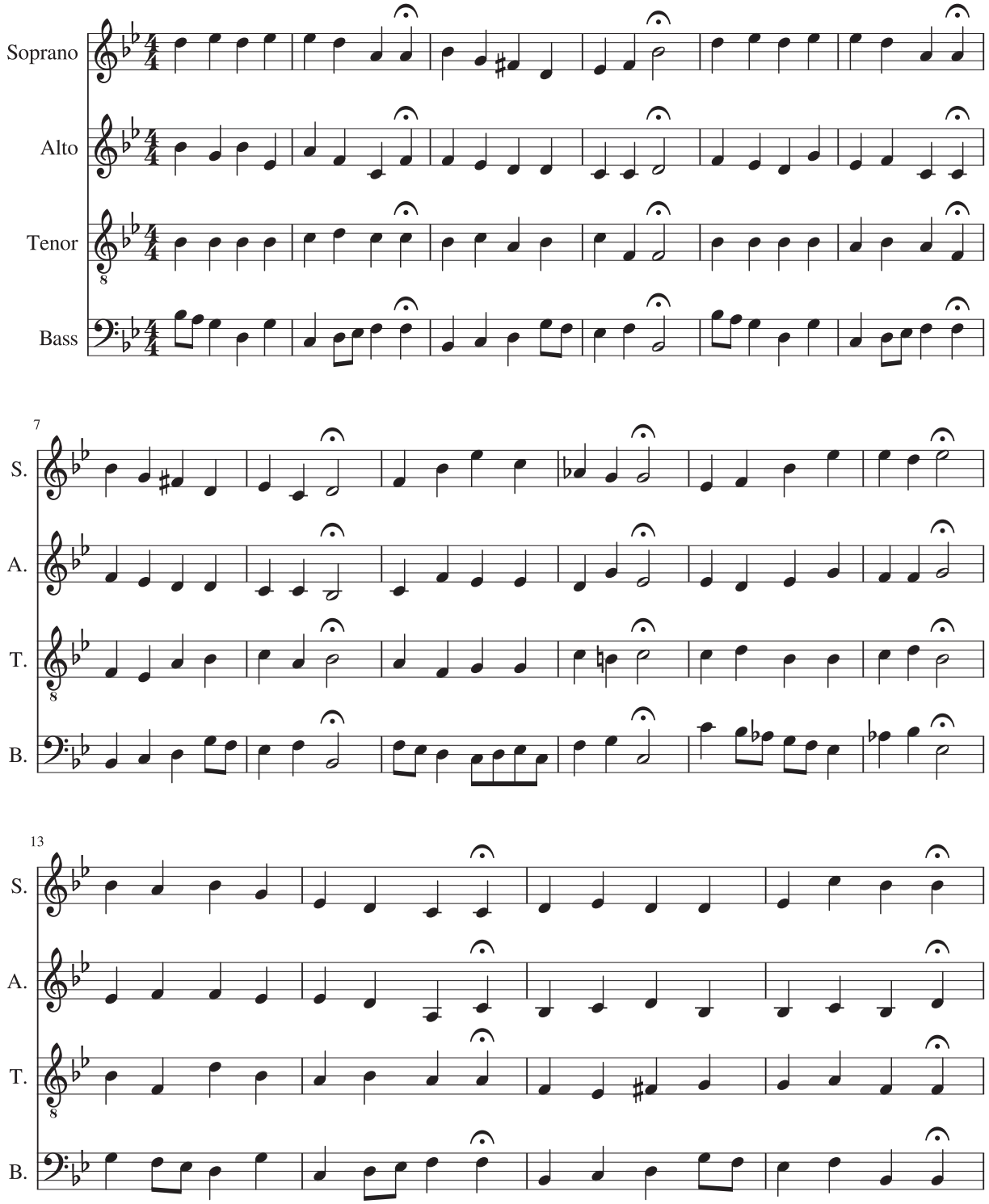


Figure 5. The automated harmonization of the bass line from Bach's chorale "O Gott, du frommer Gott." BWV 398. Sound Example 1.