SONG2QUARTET: A SYSTEM FOR GENERATING STRING QUARTET COVER SONGS FROM POLYPHONIC AUDIO OF POPULAR MUSIC

Graham Percival, Satoru Fukayama, Masataka Goto
National Institute of Advanced Industrial Science and Technology (AIST), Japan
graham@percival-music.ca, s.fukayama@aist.go.jp, m.goto@aist.go.jp

ABSTRACT

We present Song2Quartet, a system for generating string quartet versions of popular songs by combining probabilistic models estimated from a corpus of symbolic classical music with the target audio file of any song. Song2Quartet allows users to add novelty to listening experience of their favorite songs and gain familiarity with string quartets. Previous work in automatic arrangement of music only used symbolic scores to achieve a particular musical style; our challenge is to also consider audio features of the target popular song. In addition to typical audio music content analysis such as beat and chord estimation, we also use time-frequency spectral analysis in order to better reflect partial phrases of the song in its cover version. Song2Quartet produces a probabilistic network of possible musical notes at every sixteenth note for each accompanying instrument of the quartet by combining beats, chords, and spectrogram from the target song with Markov chains estimated from our corpora of quartet music. As a result, the musical score of the cover version can be generated by finding the optimal paths through these networks. We show that the generated results follow the conventions of classical string quartet music while retaining some partial phrases and chord voicings from the target audio.

1. INTRODUCTION

Cover songs are arrangements of an original song with certain variations which add novelty. Changing the instruments used is one such variation, but a complete switch of instrumentation may result in very unusual parts. For example, completely replacing a chord-heavy guitar part with a violin may result in unplayable (or very difficult) chords. Arranging music for different instruments requires consideration about the music those instruments normally perform.

Previous approaches in automated arrangement are mostly performed in the symbolic domain of music. Melody harmonization and re-harmonization of chord sequences take symbols of chords or pitches as inputs [1, 7, 10, 16]. Guitar arrangements of piano music can be generated from a MusicXML score [14]. Statistical modelling of a corpus has also been used to generate electronic dance music [6]. Furthermore, automatically generating music in a specific instrumental style is not well explored. In a great deal of work on computer-assisted composition [8], some automatic composition systems attempted to generate results with a particular composer's musical style [4] or the user's musical style [15]. However, those systems cannot be used to generate cover songs in a particular instrument style by preserving the recognizable parts of the original songs.

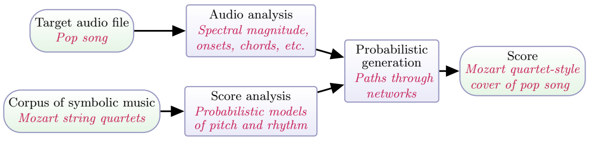


Figure 1: Generating a cover song with a specific style. Sample results are available at:
https://staff.aist.go.jp/m.goto/Song2Quartet/

We present Song2Quartet to address this issue. An overview of our system is shown in Figure 1. Two novel aspects of this work, the audio analysis for generating cover songs and generating music in a specific instrumental style, are addressed in the audio analysis and score analysis modules, respectively.

To ensure that the generated cover songs include features that are also recognizable in the original audio, the audio analysis module estimates notable rhythms, chord voicings, and contrary motions between melody and bass by extracting the audio spectrum. In parallel, to generate music to be playable and recognizably following the classical string quartet style, the score analysis module captures characteristics of the string quartet from the corpus of symbolic music such as the typical note onsets in a measure and the pitch transitions of each instrument in the quartet.

These two aspects are balanced by means of a probabilistic formulation, where the corpus style and audio analysis are combined by weighted multiplication. The audio analysis provides probabilities for observing note events at every 16^{th} note, and the score analysis mainly provides the transition probabilities of notes. We formalize our generation of cover songs as finding the sequence of notes which maximizes the probabilities obtained from the modules using dynamic programming, with techniques to compress the search space to make our problem tractable.

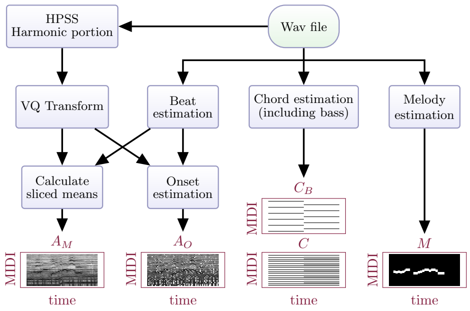


Figure 2: Audio analysis (4 measures shown in examples).

2. ANALYSIS

2.1 Features needed from audio

Knowing which pitches are in the polyphonic music is useful in creating cover songs. Since multi-pitch analysis methods often suffer from pitch and onset detection errors when handling polyphonic music with a drum track, we cannot simply apply the analysis beforehand and use the analysis results as constraints. However, we can use the audio feature extraction portions of multi-pitch analysis to aid in generating cover songs. Concretely, after performing Harmonic/Percussive source separation, the magnitudes and onsets of each note are obtained by applying a variable-Q spectral transform and calculating the salience function of the onset events.

The melody, chords, bass, and beats of a song provide musical facets which should be observed in the cover version of a song. These facets are extracted from the audio using Songle, a music understanding engine [13]. The melody and bass pitches, as well as the chord labels, are segmented according to the time grid provided by the analyzed beats. Later, these will be combined with the beat-aligned audio spectral analysis to form probabilistic constraints.

2.2 Audio analysis

Figure 2 shows an overview of the audio analysis. We perform Harmonic/Percussive source separation with median filtering [9], then use a variable-Q transform (VQT) [18] with a Blackman-Harris window and the variable-Q parameter γ set to use a constant fraction of the equivalent rectangular bandwidths [11], giving us spectral analysis S. The frequency range was set to 65 Hz–2500 Hz (MIDI pitches 36–99).

We then perform beat estimation on the original audio with Songle and divide each beat into 4, giving 16th notes. The means (over time) of VQT bins that fall within the range of each 16th note are calculated, producing the sliced spectrogram A_M. A_M is normalized to the range $[0, 1]$.

To estimate onset probabilities in the target song, we use two methods: flux of A_M and first-derivative Savitzky-Golay filtering [17] on S. The flux of A_M is simply the half-wave rectified difference between successive 16th notes of A_M. For the latter method, we calculate the smoothed first derivative of S along the time axis using Savitzky-Golay filtering with a window size of 21 samples to find the peaks of S. To quantize the onset estimation to the 16th-note level, we find the maximum peak within a time window equal to a 16th note duration, but shifted backwards in time by 25% to accommodate slight inaccuracies in the beat detection. Both methods operate on each MIDI pitch independently. We set A_O to be the sum of the two methods, and normalize it to the range $[0, 1]$.

Finally, we extract two more pieces of information using [13]: the melody M, and the chords in each song, including both the overall chord name C and the bass pitch C_B.

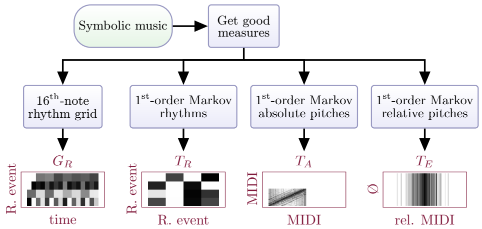


Figure 3: Score analysis (Mozart cello in examples).

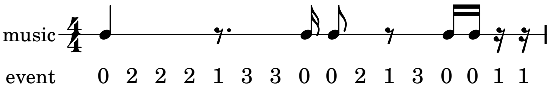


Figure 4: Rhythmic events detected in the score.

2.3 Features needed from the score

Features obtained from the score analysis contribute to maintaining the musical style. Classical string quartet music rarely includes complex rhythms and very large pitch intervals, so we obtain these tendencies as probabilities of rhythm and pitch intervals from the corpus of scores.

2.4 Score analysis

Figure 3 shows an overview of the score analysis. We used the Music21 [5] toolkit and corpus to analyze string quartets by Haydn, Mozart, and Beethoven. Our analysis comprised of pitches and rhythms, and only used music in 4/4 time which fit into a 16th-note grid. If the time signature changed in the middle of a movement, we only considered the portion(s) in 4/4.

We calculated the probabilities of rhythmic events in a 16th note grid. Rhythmic events were defined as one of four possible values: 0 indicated a new note, 1 indicated a new rest, 2 indicated a continued note, and 3 indicated a continued rest; an example is shown in Figure 4. This resulted in a 4x16 matrix of probabilities G_R, with each probability being the number of occurrences divided by the number of measures.

We extracted 1st-order Markovian [2] rhythm transitions. This is simply the probability of each [previous event, next event] pair occurring, and produced a 4x4 matrix T_R.

We calculated 1st-order Markovian pitch transitions for both absolute pitches and relative pitches. We considered a chord-note or pair of chords to include every pitch transition between the notes in successive chords. For simplicity, we recorded these transitions in two 100x100 matrices T_A

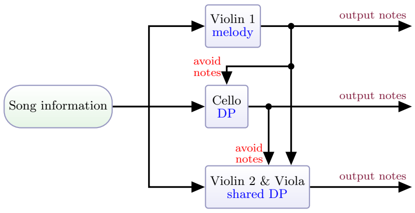


Figure 5: Overview of creating parts.

and T_E, even though a classical string quartet will not have any notes below MIDI pitch 36. For the absolute pitches, we added a 10^{-3} chance of any transition between valid pitches; this is necessary to allow some modern pop songs with non-classical chord progressions to be generated, particularly in the cello which is limited to the bass notes C_B.

3. PROBABILISTIC GENERATION

Figure 5 gives an overview of generating the quartet parts. First, the violin 1 part is set to the melody. Second, the cello part is generated with a probabilistic method and dynamic programming. Third, the violin 2 and viola parts are generated together via the same probabilistic method and dynamic programming.

To prepare for the dynamic programming, we need to define the *emission* and *transition* matrices, denoted by E and T, respectively. Our time unit is 16th notes, and we consider 200 possible events for each time-slice: 0 is a rest, 1–99 are note onsets of the same MIDI pitches, 100 is a held rest, and 101–199 are held notes (of MIDI pitch +100). We define N as the number of 16th notes in the target song. An overview of calculating E and T is shown in Figure 6.

3.1 Constructing probability matrices

3.1.1 Construction emission probabilities E

The emission probabilities E is a matrix of size $N \times 200$, representing every possible event at every 16th note. They are generated by calculating E_O (onsets) and E_H (held notes), each of size $N \times 100$,

$$E_O = A'_O \otimes C' \otimes G'_R \otimes I_R \otimes V \tag{1}$$

$$E_H = A'_M \otimes C' \otimes G'_R \otimes I_R \otimes V \tag{2}$$

where $\otimes$ is the element-wise product. The intuition behind this multiplication is that we consider each variable to be an independent probability distribution, so we are calculating the joint distribution. E_O and E_H are then stacked vertically to form E. The variables are:

- A'_O, A'_M — *Audio onsets and magnitudes*: Audio onsets A_O and magnitudes A_M for MIDI pitches 1–99 are taken directly from the audio analysis. The “silence” event (0) is set to a constant value of 10^{-5}.
- C' — *Chord tones*: We construct a matrix of all MIDI pitches for every 16th note in the song; each cell is 1 if that pitch is in the given chord, 10^{-2} otherwise. For the cello, we use the bass note of each chord C_B; for other instruments, we use any chord tone included in C.

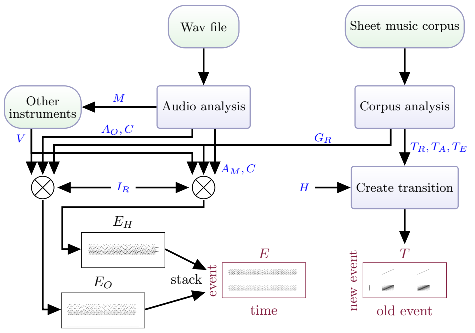


Figure 6: Calculating emission and transition probabilities E and T. $\otimes$ indicates element-wise multiplication.

- G'_R — *Rhythm grids*: We take the overall probability of a rhythmic event in the corpus at each 16th note G_R, and repeat it for every 16 time-slices in N.
- I_R — *Extreme instrument ranges*: We specify maximum bounds for instrument ranges: MIDI pitches 36–69 for cello, 48–81 for viola, and 55–99 for violin. When a corpus of symbolic music is used, the pitch transition probabilities narrow these ranges; I_R is only relevant if the user chooses not to use any corpus.
- V — *Avoid previously-used notes*: We reduce the probability of using the same notes as other instruments by setting them to 10^{-2} in V; other events are set to 1. We also reduce the probability of playing a note one octave higher than an existing note (as those are likely octave errors in the audio analysis) by likewise setting those values to 10^{-2}.

We eliminate any non-rest values less than 10^{-3} to reduce the computational load for music generation.

3.1.2 Construction transition probabilities T

The transition probabilities T are a matrix of size 200×200, representing every possible event-to-event pair.

$$T = T'_R \otimes T'_A \otimes T'_E \otimes H \tag{3}$$

The variables are:

- T'_R — *Rhythm transitions*: We use T_R, the probability of each rhythm event following a previous rhythm event. The note onset and held note probabilities are copied to vectors 1–99 and 101-199 respectively, while the rest onset and held rest probabilities are copied into vectors 0 and 100.
- T'_A, T'_E — *Pitch transitions*: We use the probabilities of each pitch following a previous pitch considering absolute or relative pitches, T_A and T_E respectively. These matrices are originally 100×100; we simply copy the matrices four times to create 200×200 matrices (that is to say, allowing these relative transitions to apply to onset-onset, onset-held, held-onset, held-held pairs).
- H — *Hold-events only after onset-events*: Each “hold” event (events 100 and up) can only occur after its respective “onset” event. We formalize this constraint as a matrix H where rows 0–99 are all 1, while rows 100–199 contain two identity matrices (in columns 0–99 and 100-199).

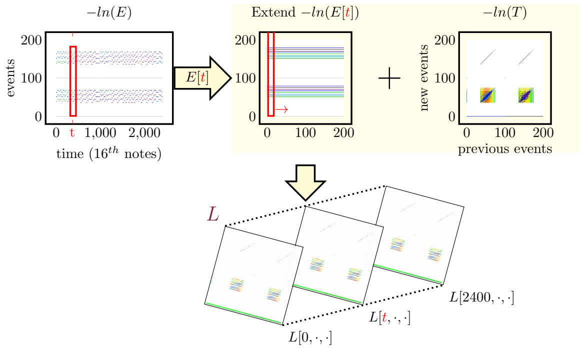


Figure 7: Combining emission and transition probabilities E and T into overall log-probabilities L.

3.2 Generation of a cover song under constraints

We combine E and T to form the log-probability L of the arranged score given the observed audio and corpus data, which has dimensions $N \times 200 \times 200$. For each $t \in N$,

$$L[t,\cdot,\cdot] = -\ln(E[t,-,:]) - \ln(T) \qquad (4)$$

where $E[t,-,:]$ indicates that the 1x200 column vector $E[t]$ is extended to form a 200x200 matrix. This is illustrated in Figure 7. Since E and T contain very small numbers, we add their negative log-values instead of multiplying them.

L can be visualized by considering it to be a network of time-events (Figure 8). The maximum probability of a given score occurs when the negative log-probability is minimized; i.e. by finding the shortest path through L with a standard dynamic programming algorithm [3].

3.2.1 Local and Global Shortest Paths

As shown in Figure 5, we calculate the cello accompaniment part first. After that, we could solve the viola and then violin 2 parts separately, but we found that this occasionally produced very high violin 2 music. Instead we solve the violin 2 and viola parts together, with the constraint that they cannot play the same pitch at the same time.

In order to find two shortest paths simultaneously, we construct a large network with every possible combination of nodes from each time-slice of the individual violin 2 and viola networks. For example, if at time t the violin 2 could have 4 possible events and the viola could have 5 possible events, then the combined network will have 20 possible events for time t. The edge weights are simply the sum of the existing edges from the individual networks.

3.2.2 Compacting Matrices

To lower memory usage and improve processing time, we reduce the size of the matrices. We construct a mapping for each time-slice t between the non-infinite weights in L and a smaller matrix. This takes approximately 1 second, and results in a matrix which is roughly 1% of the original size (e.g., 96 million entries reduced to 1.2 million entries). Note that this compression is lossless and does not affect the shortest-path calculation, as an edge with weight ∞ will not appear in the shortest path.

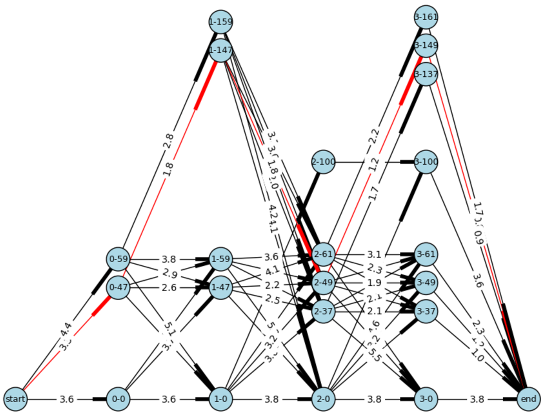


Figure 8: Network of possible pitches L; shortest path colored red. Node labels are in the form “time-event”, with event x being a MIDI pitch onset ($x < 100$) or hold ($x \geq 100$). For legibility, edges with a weight of infinity and nodes with no non-infinite-weight edges are not displayed.

“Compacting” L in this way speeds up the computation of the single cello part, but its true value is found when combining the violin 2 and viola parts. Without any compacting, a normal pop song (150 measures) produces a network for a single part with $2400 \times 200 \times 200 = 9.6 \cdot 10^7$ entries. However, naively combining the violin 2 and viola parts produces a network with $2400 \times 200^2 \times 200^2 = 3.8 \cdot 10^{12}$ entries (15 TB of memory). We therefore perform two rounds of compacting; before and after combining the parts. After compacting the individual violin 2 and viola parts, we are left with networks of size approximately 1.6 million and 2.3 million. After performing the second round of compacting (this time on the combined matrix), the memory requirement is reduced from 5.8 GB to 0.25 GB.

3.2.3 Weighted probabilities

We found that the initial system produced music which was too heavily biased towards one “prototypical” measure of rhythms for each composer. We therefore multiplied each matrix by its own weighting factor, and allowed the user to specify and experiment with their own desired weights.

4. EXAMPLES AND DISCUSSION

To illustrate aspects of the generated music, we created a few cover versions of “PROLOGUE” (RWC-MDB-P-2001 No.7) from the RWC Music Database [12], with a variety of weights to the probability distributions. Short excerpts of the beginning of “PROLOGUE” are shown in Figure 10 with four variations: no corpus analysis, no audio spectral analysis, equal weights, and a set of custom weights.

Figures 10a and 10b clearly demonstrate the usefulness of combining audio with score analysis. Figure 10a does not use any corpus information (the weights of G_R, T_R, T_A, and T_E are set to 0), and produces music which is not idiomatic and is extremely difficult to perform. In the other extreme, Figure 10b uses the full Haydn string quartet corpus analysis, but does not use any spectral information (the weights of A_O and A_M are set to 0), and produces music which is playable but very repetitive and "boring": Other than measure 10 (the transition from the introduction to the melody), each instrument in the accompaniment plays the same rhythm in every measure (with the exception of the cello in measure 5), and 76% of measures contain a single pitch while 24% of measures contains two pitches.

Figure 10c uses all available data with weights of 1, and the music is both quite playable and more interesting than Figure 10b. There is more variation in the rhythms, and most notes are typical classical-style durations such as whole notes, half notes, or quarter notes. There are a few non-chord pitch changes (e.g., violin 2 measure 3, viola measure 13), but not many. This version contains one mistake: the viola in measure 13 begins with a C♮ 16th note which quickly changes to a $C\sharp$ chord-tone. This could be avoided by decreasing the probability of non-chord tones, but doing so would also decrease the chance of a non-chord tone in the original song from being reproduced. This is an illustration of the choices available to the user.

Figures 10d (Haydn), 10e (Mozart), and 10f (Beethoven) demonstrate a custom set of weights. After some experimentation, we (subjectively) chose to set the onset A_O weight to 0.9, the corpus rhythms G_R and T_R weights to 0.5, and the corpus pitch transition T_A and T_E weights to 0.25. These three cover versions produce noticeably distinct music, arising solely due to the corpus used. The overall distribution of rhythmic durations seems natural: the cello has longer notes than the inner two voices. The distribution of pitches is reasonable, with all instruments playing in a comfortable range; the corpus clearly helps in avoiding the extreme pitches that were present in Figure 10a.

A few parts of the cover versions are the same in all compositions. Measure 10 always ends with a G♯-C♮ (alternatively "spelled" as B♯) in the cello and violin 2, with the viola filling in a transition from D♯ to C♮ (or B♯); this makes a nice V–I chord sequence (G♯ major to C♯ major) leading into measure 11. In addition, the V–I resolution in measures 10–11 always includes contrary motion in the cello and violin 2. Our probabilistic generation does not take relative motion of multiple voices into account, so this nice voice leading must arise from the strength of its presence in the audio spectral analysis.

A few problems exist in the voice leading. For example, Figure 10d shows a number of parallel fifths (e.g., viola and cello, measures $4 \rightarrow 5 \rightarrow 6$, $8 \rightarrow 9$). These likely arise due to the 2nd and 3rd harmonics of bass guitar notes in the original recording. A similar problem occurs with sudden jumps of an octave after one 16th or 8th note appears in a few places (e.g., viola measure 4 and cello measure 12). These also likely arise due to inaccuracies in the spectral analysis: the energy in upper partials of a single note can vary, so multiple onsets are detected in close succession. More advanced signal processing in terms of onset estimation or pitch salience calculation could mitigate this issue. Another fix for the parallel fifths would be to use a more advanced mathematical model; a first-order Markov model does not track the inter-dependence between quartet parts.

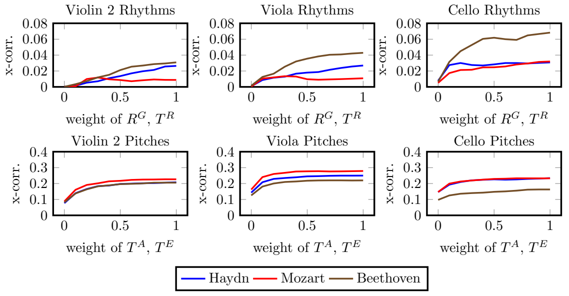


Figure 9: Objective analysis of weights; unless otherwise specified, our custom weights are used.

4.1 Objective analysis

Figure 9 shows the effect of changing the rhythmic or pitch corpus weights. The "pitch" plots show the cross-correlation between the corpus relative pitch distribution T_E and the relative pitches calculated from the generated scores. The "rhythm" plots show the cross-correlation between corpus and generated scores, based on the types of measures appearing in the output. Concretely, we construct a dictionary of full-measure rhythmic events (such as 0222133002130011 from Figure 4) along with their frequency of appearance, for both the corpus and the generated music. We then calculate the cross-correlation between those dictionaries for the corpus and each cover version.

Increasing the weight generally increases the correlation between corpus and generated music for both pitches and rhythms. One counter-example is violin 2 and viola in Mozart quartets. We theorize that this arises because increasing the rhythmic weight reduces the number of "completely eighth note" measures in the generated music, however such measures are very common in the original corpus.

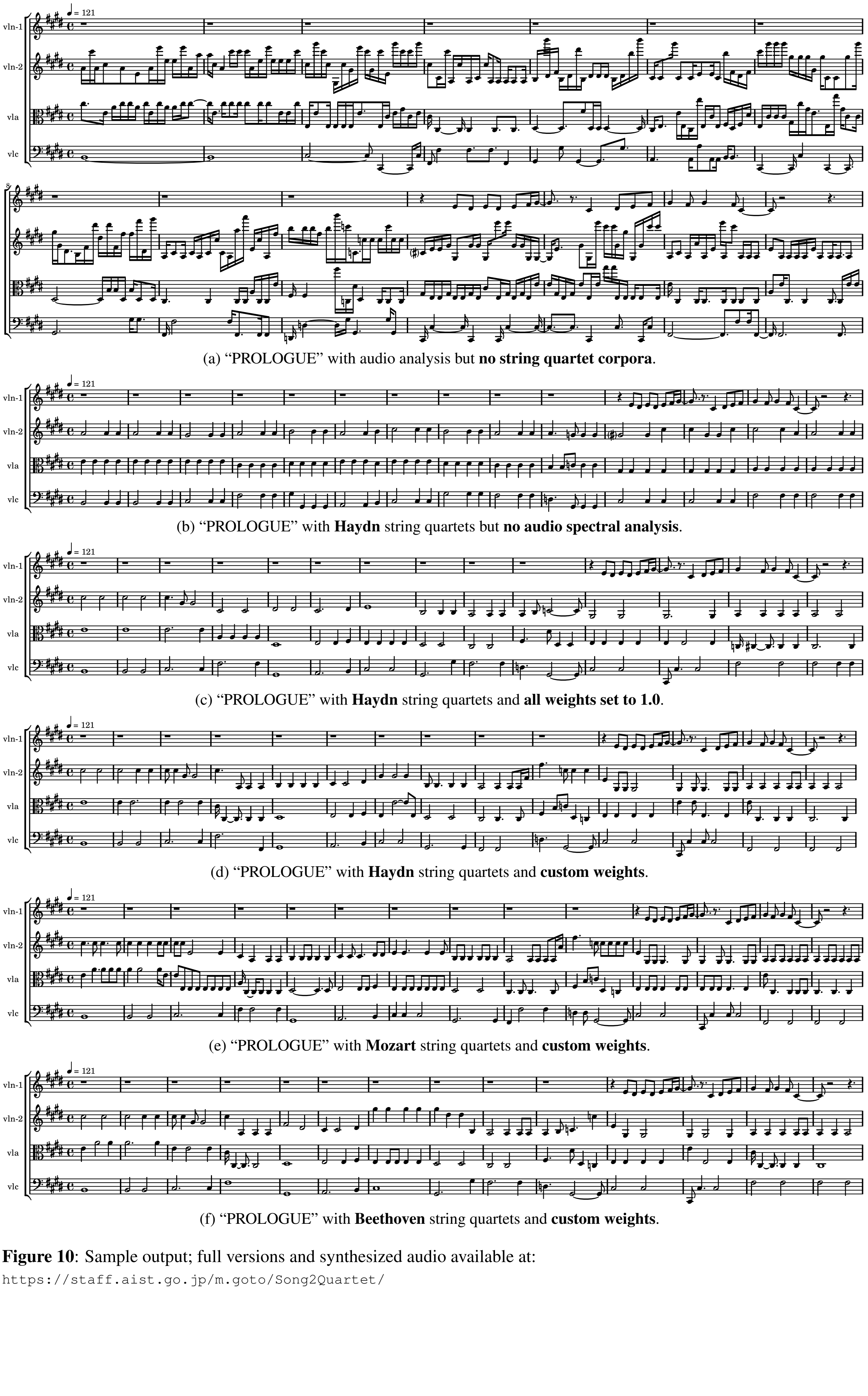
5. CONCLUSION AND FUTURE WORK

We presented Song2Quartet, a system for generating string quartet cover versions of popular music using audio and symbolic corpus analysis. Both the target pop song audio file and the corpus of classical music contribute to the output; using only one or the other produces clearly inferior results. In order to avoid awkward second violin parts, we performed a semi-global optimization whereby we created the second violin and viola parts at the same time.

The current system makes a number of ad hoc assumptions, such as the melody always being played by the first violin and all rhythms fitting into 16th-note rhythms. Our evaluation was primarily based on informal listening, which showed promise despite some voice leading errors.

We plan to extend the data-driven corpus analysis so that users may generate cover versions for other groups of classical instruments. We also plan to add a GUI so that users can place the melody in different instruments at any point in the song. Finally, we would like to include evaluations of the generated scores' "playability" by musicians.

Acknowledgments: This work was supported in part by CREST, JST.



(a) "PROLOGUE" with audio analysis but **no string quartet corpora**.

(b) "PROLOGUE" with **Haydn** string quartets but **no audio spectral analysis**.

(c) "PROLOGUE" with **Haydn** string quartets and **all weights set to 1.0**.

(d) "PROLOGUE" with **Haydn** string quartets and **custom weights**.

(e) "PROLOGUE" with **Mozart** string quartets and **custom weights**.

(f) "PROLOGUE" with **Beethoven** string quartets and **custom weights**.

Figure 10: Sample output; full versions and synthesized audio available at: `https://staff.aist.go.jp/m.goto/Song2Quartet/`

6. REFERENCES

[1] Moray Allan and Christopher K. I. Williams. Harmonising chorales by probabilistic inference. In *NIPS*, pages 25–32, 2005.

[2] Charles Ames. The markov process as a compositional model: a survey and tutorial. *Leonardo*, pages 175–187, 1989.

[3] Richard Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, USA, 1 edition, 1957.

[4] David Cope. Computer modeling of musical intelligence in EMI. *Computer Music Journal*, pages 69–83, 1992.

[5] Michael Scott Cuthbert, Christopher Ariza, and Lisa Friedland. Feature extraction and machine learning on symbolic music using the music21 toolkit. In *ISMIR*, pages 387–392, 2011.

[6] Arne Eigenfeldt and Philippe Pasquier. Considering vertical and horizontal context in corpus-based generative electronic dance music. In *Proceedings of the Fourth International Conference on Computational Creativity*, volume 72, 2013.

[7] Benjamin Evans, Satoru Fukayama, Masataka Goto, Nagisa Munekata, and Tetsuo Ono. AutoChorusCreator: Four-Part Chorus Generator with Musical Feature Control, Using Search Spaces Constructed from Rules of Music Theory. In *International Computer Music Conference*, 2014.

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

[9] Derry Fitzgerald. Harmonic/Percussive Separation using Median Filtering. In *International Conference on Digital Audio Effects*, 2010.

[10] Satoru Fukayama and Masataka Goto. Chord-sequence-factory: A chord arrangement system modifying factorized chord sequence probabilities. In *ISMIR*, pages 457–462, 2013.

[11] Brian R Glasberg and Brian CJ Moore. Derivation of auditory filter shapes from notched-noise data. *Hearing research*, 47(1):103–138, 1990.

[12] Masataka Goto, Hiroki Hashiguchi, Takuichi Nishimura, and Ryuichi Oka. RWC Music Database: Popular, Classical and Jazz Music Databases. In *ISMIR*, volume 2, pages 287–288, 2002.

[13] Masataka Goto, Kazuyoshi Yoshii, Hiromasa Fujihara, Matthias Mauch, and Tomoyasu Nakano. Songle: A web service for active music listening improved by user contributions. In *ISMIR*, pages 311–316, 2011.

[14] Gen Hori, Hirokazu Kameoka, and Shigeki Sagayama. Input-Output HMM Applied to Automatic Arrangement for Guitars. *Journal of Information Processing*, 21(2):264–271, 2013.

[15] Francois Pachet. The continuator: Musical interaction with style. *Journal of New Music Research*, 32(3):333–341, 2003.

[16] Francois Pachet and Pierre Roy. Musical harmonization with constraints: A survey. *Constraints*, 01/2001(6):7–19, 2001.

[17] Abraham. Savitzky and M. J. E. Golay. Smoothing and differentiation of data by simplified least squares procedures. *Analytical Chemistry*, 36(8):1627–1639, 1964.

[18] Christian Schörkhuber, Anssi Klapuri, Nicki Holighaus, and Monika Dörfler. A matlab toolbox for efficient perfect reconstruction time-frequency transforms with log-frequency resolution. In *Audio Engineering Society Conference: 53rd International Conference: Semantic Audio*. Audio Engineering Society, 2014.