STYLE-SPECIFIC ACCOMPANIMENT GENERATION VIA CONSTRAINED DISCRETE OPTIMIZATION

Stephanie Y. H. LEW[1], **Wee Kheng LEOW**[2], and **Kat R. AGRES**[3]

[1,2]*Department of Computer Science*, **National University of Singapore**, Singapore
[3]*Centre for Music and Health, Yong Siew Toh Conservatory of Music*, **National University of Singapore**, Singapore

ABSTRACT

Style-specific accompaniment generation aims to produce accompaniment for a given melody in a target style. Despite the success of recent methods like deep neural networks (DNNs) and hidden Markov models, they have several shortcomings. To adapt to different styles, they require separate training on style-specific datasets, increasing total training time. Training on mixed-style datasets can result in diluted, inconsistent outputs. Moreover, they cannot handle section closure and global music structure effectively. The proposed method, FlexFlow, overcomes the shortcomings of existing methods by formulating style-specific accompaniment generation as a constrained discrete optimization problem. It defines style as the pitch progression in user-specified reference phrases, enabling adaptation to any target style expressed in those references. It can generate accompaniment with as little as one reference phrase. It adopts a training-free approach that adjusts individual pitches to satisfy chord constraints while preserving pitch direction and pitch difference of reference phrases. It can handle section closure, contrast and phrase repetition to match the structure of full-length melodies. Performance tests and a user study indicate that FlexFlow's accompaniments are more harmonious than those of comparable DNNs and its quality is closer to human-composed accompaniments compared to that of DNNs.

1. INTRODUCTION

Style-specific accompaniment generation aims to generate accompaniment to a given melody in a target style. This problem is important in automatic music generation and has practical applications in video, games and music therapy. Prior works on accompaniment generation tend to focus on harmonic and rhythmic fit to a given melody, but not style-specificity. Style determines the overall character of a music piece and can refer to a genre or a composer. The same melody can evoke different moods when accompanied in pop or jazz styles. Unlike harmony or rhythm, style is a multi-level property that describes notes, phrases, and sections [1, 2], which is difficult to model and control.

Recent accompaniment generation methods include hidden Markov models (HMMs) [3] and deep neural network (DNN)-based methods like GANs, Transformers, diffusion models and VAEs. Existing GANs generate single or multi-track accompaniment for leadsheets [4–6] and Transformers generate polyphonic piano and multi-instrument accompaniment for melodies with long-term structure [7, 8]. Diffusion models generate accompaniment hierarchically from global structure to pitches [9]. VAE and HMM have been combined to generate accompaniment phrase-wise for complete melodies [10]. These methods are trained on a large training set of a specific genre such as pop or jazz. To accommodate different styles, they require separate training or fine-tuning on style-specific datasets, increasing the total training time. If they are trained on a dataset containing various styles, they may produce stylistically diluted or inconsistent results.

To address the shortcomings of existing methods, this paper formulates style-specific accompaniment generation as a constrained discrete optimization problem. The proposed method, FlexFlow, enables style control using one or more user-specified reference phrases, with accompaniment style represented by their pitch progressions. It generates style-specific accompaniment by selecting reference phrases and transforming them to fit the target melody and its associated chords. It also handles section closure and phrase repetition to match the melody's global structure.

The paper is organized as follows: Section 2 describes the problem formulation and Section 3 presents the proposed algorithm. Section 4 reports test results, including user study on accompaniment quality. The contributions are:

1. Achieve style-specificity and style control using as little as a single reference accompaniment phrase.
2. Generate accompaniment in various styles.
3. Apply direct optimization that does not require training on a large training dataset.
4. Handle section closure that is absent in prior works.
5. Generate accompaniment for full-length melodies that align with global music structure.

2. PROBLEM FORMULATION

This section formulates style-specific accompaniment generation as a constrained discrete optimization problem. It covers key design aspects of FlexFlow, including pitch progression, accompaniment-chord matching, accompaniment-melody matching, phrase closure, and phrase repetition.

2.1 Accompaniment Style

Melody is represented as a step function of pitches p_i over integer time index $i = 0, \ldots, m-1$, at sixteenth note resolution. Each pitch p_i is a MIDI pitch $p_i \in \{0, 1, \ldots, 127\}$. Pitch onsets and rests are represented separately as rhythmic pattern. The melody is organized into bars that contain fixed length fragments $h_j, j = 0, \ldots, n-1$. In $2/4$ and $3/4$ time, each bar contains a single fragment. In $4/4$ and $6/8$ time, each bar contains two fragments.

Let F denote a reference accompaniment phrase provided by the user. It contains η fragments f_l, for $l = 0, \ldots, \eta-1$ where $\eta \in \{1, 2, 4, 8, 16, \ldots\}$. A reference fragment f_l has the same length as a melody fragment h_j and a target accompaniment fragment g_j. Reference fragments are related to target fragments by:

$$l = j \bmod \eta. \tag{1}$$

For example, if $\eta = 2$, then g_0, g_2, g_4, etc. use f_0, and g_1, g_3, g_5, etc. use f_1. For melodies with contrasting sections (e.g., A–B–A), distinct reference phrases F_a and F_b can be provided by the user to introduce section contrast.

2.2 Pitch Progression

In this work, pitch progression is modelled as a key feature for representing the style of a reference accompaniment fragment. Based on prior pitch representations [11, 12], pitch progression is described at two levels: detailed property and overall property. Detailed property of pitch progression consists of pitch direction and pitch difference. Let p_i denote a MIDI pitch in reference accompaniment fragment f_l. Then, its pitch direction is $\text{sign}(p_i - p_{i-1})$ and pitch difference is $|p_i - p_{i-1}|$. Suppose reference fragment f_l corresponds to target fragment g_j. To preserve the detailed property of pitch progression, the pitches in fragment g_j should follow the same pitch directions and similar pitch differences as those in fragment f_l.

Overall property characterizes the shape of pitch contour, which can be captured by the Discrete Cosine Transform (DCT) of the pitch sequence [12]. The first DCT coefficient represents the overall pitch level of the pitch contour. The next three DCT coefficients form a 3D vector that represents the overall shape of pitch contour that is independent of the overall pitch level. The difference in DCT vectors is used to measure the consistency of generated accompaniment fragments in the same melody section.

2.3 Accompaniment-Chord Matching

Matching accompaniment to the chord progression is important for achieving harmonic coherence. In this work, accompaniment-chord matching is modelled as a design choice that encourages most, if not all, accompaniment pitches to be chord pitches, with the goal of reinforcing the harmonic structure of the accompaniment. Non-chord pitches, such as passing or suspension notes, may be used sparingly to introduce tension or variation. Excessive use can weaken harmonic coherence and disrupt the relationship between accompaniment and melody [13, 14]. In FlexFlow, accompaniment-chord matching is achieved by ensuring that all pitches of accompaniment fragment g_j are chord pitches of their associated chord c_j. Non-chord pitches are allowed by accompaniment-melody matching.

2.4 Accompaniment-Melody Matching

Music perception studies suggest that the perception of simultaneously sounding pitches is influenced by pitch dissonance [15]. Accordingly, the match between an accompaniment fragment g_j and a melody fragment h_j is measured by the dissonance between their pitches. Given two pitches p and q, the dissonance $d(p, q)$ is a function of their pitch interval $\theta(p, q) = |p - q| \bmod 12$ [16]. The value of $d(p, q)$ ranges from 0 to 1. Specifically, $d(p, q) = 0$ when $\theta(p, q) = 0$, $d(p, q) = 1$ when $\theta(p, q) = 1$ or 11, and $0 < d(p, q) < 1$ for $2 \leq \theta(p, q) \leq 10$. The table of dissonance values for all pitch intervals is provided in [17].

A similar concept can be defined for the dissonance $d(h_j, g_j)$ between melody fragment h_j and accompaniment fragment g_j. It is measured according to the dissonance between the pitches p_i of melody fragment h_j and the pitches r'_i of accompaniment fragment g_j:

$$d(h_j, g_j) = \sum_i d(p_i, r'_i), \text{for } p_i, r'_i \neq \text{rest}. \tag{2}$$

2.5 Closure and Repetition

Phrases or sections in a melody often end with sustained pitches, providing a sense of closure similar to punctuation [18]. Similarly, this work introduces sustained pitches at the end of selected phrases and sections to provide a sense of closure. The durations of the sustained pitches should match those in the melody. The melody fragments h_j at phrase or section endings that require closure are labeled with structure label $\lambda_j =$ 'C'. The structure label λ_j also plays another role. When λ_j takes on an integer value j', it indicates that melody fragment h_j repeats $h_{j'}$ exactly.

3. ALGORITHM DESIGN

The proposed algorithm, FlexFlow, consists of two stages: (1) reference phrase selection and (2) pitch adjustment. It selects reference accompaniment phrases that match target melody phrases according to melody structure, and adjusts their pitches by minimizing the overall dissonance D between the target accompaniment fragment g_j and the melody fragment h_j:

$$D = \sum_{j=0}^{n-1} d(h_j, g_j). \tag{3}$$

To capture the style of the reference accompaniment, FlexFlow keeps the pitch directions in the target accompaniment identical to those in the reference accompaniment, and minimizes their dissimilarity in pitch differences. For closure, it applies sustained pitches at selected phrase and section endings.

3.1 FlexFlow Accompaniment Generation

Algorithm 1 shows how FlexFlow chooses reference accompaniment fragments f_l and transforms them to generate the corresponding target accompaniment fragments g_j.

Algorithm 1: FlexFlow Accompaniment Generation

Input: Melody M, chords c_j, reference accompaniment phrases $\{F\}$.
Output: Target accompaniment G.

1 Determine the accompaniment octave.
2 **for** $j = 0, \ldots, n-1$ **do**
3 Select the reference accompaniment phrase F that matches the section label of melody fragment h_j.
4 Select the reference accompaniment fragment f_l in F that corresponds to melody fragment h_j.
5 Transpose reference fragment f_l to target fragment g_j by matching the key of melody M and chord c_j.
6 Adjust selected pitches in g_j to nearest desired pitches while retaining pitch directions.
7 **if** $\lambda_j =$ 'C' **then**
8 Adjust the last pitch onset and its duration in target fragment g_j, according to those in the melody fragment h_j.
9 **else if** λ_j *is an integer* j' **then**
10 Set $g_j = g_{j'}$.

Line 1 determines the octave of the target accompaniment. Depending on the pitch ranges of the melody and the reference accompaniment, the accompaniment octave can be set at 1 or 2 octaves below the melody. The actual choice of octave is to be decided by the user.

Line 3 selects the reference accompaniment phrase F that matches the section label of melody fragment h_j. Line 4 selects from phrase F the accompaniment fragment f_l that corresponds to melody fragment h_j, according to Eq. 1.

Lines 5 and 6 performs pitch transformation to ensure harmonic coherence between the accompaniment, melody and the associated chord. Line 5 transposes reference fragment f_l to target fragment g_j by matching the key of melody M and chord c_j, in the appropriate accompaniment octave. The pitch interval θ of transposition is calculated as the difference between the lowest pitch q in chord c_j and the lowest pitch r in reference fragment f_l, adjusted to fit the accompaniment octave: $\theta = r - q + 12\gamma$, where $\gamma \geq 1$ is dependent on the pitch ranges of the melody and the reference accompaniment. Typically, $\gamma = 1$ or 2. Then, the target accompaniment pitches r'_i in fragment g_j is computed by subtracting the transposition interval θ from the reference pitches r_i, i.e., $r'_i = r_i - \theta$. Line 6 adjusts selected pitches in the target accompaniment fragment g_j to their nearest desired pitches while preserving their pitch directions. This step is performed via discrete optimization, as detailed in Section 3.2.

Lines 7 and 8 adjust the last pitch onset and its duration in target fragment g_j with structure label λ_j of 'C' to provide closure. For example, the melody always ends with a sustained pitch at the end of a song (Fig. 2). The sustained pitch, along with its onset time index, is copied to the target fragment, and the pitch level is set to the bass pitch in the associated chord c_j while retaining pitch direction.

In principle, the rhythmic pattern of the generated accompaniment can also be adjusted. However, pitch adjustment is more crucial than rhythmic adjustment in ensuring that the generated accompaniment matches the target chords and harmonically supports the melody. User study results (Section 4.3) indicate that listeners are more sensitive to dissonant harmony than to minor rhythmic variations in the accompaniment. Thus, only adjustment of rhythmic pattern for closure is considered. Other possible ways of adjusting rhythmic pattern are highlighted in Section 5.

Lines 9 and 10 handle repetition. When the structure label λ_j is an integer j', it indicates that the melody fragment h_j is an exact repetition of an earlier fragment $h_{j'}$. In this case, the accompaniment fragment $g_{j'}$ is copied to g_j.

3.2 Pitch Adjustment via Discrete Optimization

The pitch adjustment step in FlexFlow uses discrete optimization over a set of candidate pitches. Two adjustment options are available:

1. Adjust to chord pitch: Non-chord pitches r'_i with respect to the associated chord c_j are selected. Each selected pitch r'_i is adjusted to the nearest chord pitch in c_j, in the same pitch direction as its corresponding reference pitch r_i. This adjustment ensures that all adjusted accompaniment pitches are chord pitches with respect to chord c_j, satisfying the constraint stated in Section 2.3, while preserving the pitch directions and pitch differences in the reference accompaniment.

2. Adjust to reduce dissonance: Accompaniment pitches r'_i that are dissonant with the corresponding melody pitch p_i are selected. An accompaniment pitch r'_i is considered dissonant with p_i if $d(p_i, r'_i) > 0.4$. Following [17], pitch intervals with dissonance scores ≤ 0.4 including perfect fourths, fifths, and octaves, are favoured as a design choice to encourage consonant accompaniment. Each selected pitch r'_i is adjusted to the nearest pitch such that dissonance $d(p_i, r'_i) \leq 0.4$, minimizing dissonance (Eq. 2, 3) while retaining the pitch direction of its corresponding reference pitch r_i. This adjustment ensures that all accompaniment pitches are not dissonant with their corresponding melody pitches.

Pitch adjustment for Option 1 is performed as follows. A pitch r'_i may be set in one of three octaves, namely $r'_i, r'_i + 12$ and $r'_i - 12$. There is no need to consider octaves further than ± 12 because the goal is to choose the nearest pitch. A triad chord has three pitches giving 9 candidate pitches to choose from. A 7th chord has four pitches giving 12 candidate pitches. Thus, the adjustment of pitch r'_i is to find, among these candidate pitches, the one that has the same pitch direction as reference accompaniment pitch r_i, and is nearest to r'_i in terms of pitch difference. Then, pitch r'_i is changed to the chosen candidate pitch level. Pitch adjustment for Option 2 is performed in a similar way, but uses low-dissonance pitches instead of chord pitches. As pitches are discrete and the number of candidate pitches is very small, the above discrete optimization can be achieved by exhaustively searching for the pitch that satisfies the stated condition. There is no need to apply complex optimization algorithm such as simulated annealing and stochastic optimization.

Option 1 ensures that all accompaniment pitches are chord pitches regardless of the melody pitches. So, the dissonance between accompaniment and melody may be higher than 0.4. On the other hand, Option 2 ensures that the accompaniment is harmonious with the melody regardless of the associated chord. Consequently, some accompaniment pitches may be non-chord pitches. One way of balancing Options 1 and 2 is by relaxing them as follows. First, apply Option 1 to obtain only chord pitches in the accompaniment. Next, identify the accompaniment pitches, which are now chord pitches, with dissonance > 0.5 compared to their corresponding melody pitches. Then, adjust these pitches to the nearest pitches with dissonance ≤ 0.5 while preserving their pitch directions. This method retains a small number of non-chord pitches with overall dissonance ≤ 0.5, thus balancing Options 1 and 2.

3.3 Computational Complexity of FlexFlow

FlexFlow runs for exactly n iterations, one for each fragment index j, to produce n accompaniment fragments. Each iteration performs a bounded set of operations. Each pitch adjustment chooses an appropriate pitch from a small set of candidate pitches. These small numbers are independent of the number of fragments n, and can be regarded as constants. Therefore, FlexFlow runs in $O(n)$ time.

3.4 Extension to Polyphonic Accompaniment

Polyphonic music such as chorales and quartets is organized into separate lines, one for each voice [19, 20]. In contrast, FlexFlow groups all pitches occurring at the same time index into a poly-pitch. Then, a polyphonic accompaniment can be represented by a single sequence of poly-pitches, each containing one or more pitches. The central idea of extending FlexFlow to polyphonic accompaniment is to characterize each poly-pitch by a representative pitch, defined as the lowest pitch in the poly-pitch. Then, polyphonic accompaniment can be regarded as monophonic accompaniment by FlexFlow. After transposition, each poly-pitch is adjusted according to either Option 1, Option 2 or the balanced version. Other pitches in a poly-pitch are adjusted accordingly to retain their pitch intervals with the adjusted representative pitch. To avoid duplicate pitches within a poly-pitch, any candidate pitch that causes potential pitch collision is not considered for pitch adjustment. Note that if the polyphonic reference contains parallel fifths or octaves, they may also appear in the generated accompaniment, as the proposed adjustment retains pitch intervals with the adjusted representative pitch.

3.5 Comparison to Existing Methods

Before evaluating the performance of FlexFlow (FF), it is useful to discuss the similarities and differences between FF and two state-of-the-art methods, namely AccoMontage (AM) [10] and Whole Song Generation (WSG) [9]. AM is a hybrid method that combines hidden Markov model (HMM) and variational autoencoder (VAE). WSG is a series of cascaded diffusion models that generate structured music across multiple levels, including phrase structure labels, melody phrases, chords, and pitches. Both AM and WSG require training. In contrast, FF applies direct discrete optimization without the need for training.

During accompaniment generation, AM, WSG and FF adopt similar two-stage approach: (1) reference accompaniment selection, and (2) pitch adjustment. In Stage 1, AM selects accompaniment phrases from a reference set provided by the user. WSG takes a single user-specified reference phrase. FF selects a user-provided reference accompaniment phrase according to the section type of the melody phrase. In Stage 2, AM applies VAE to adjust pitches of the selected reference phrases to match the melody and its associated chords. The reference phrases with pitches adjusted are the generated accompaniment. WSG applies diffusion method to adjust the pitches of the selected reference phrase to match the melody. FF transposes the selected reference phrase according to the bass pitches of the melody chords, and then adjusts the accompaniment pitches to match the chords and, optionally, to reduce dissonance with the melody, while preserving the pitch directions and pitch differences of the reference phrase. In summary, AM, WSG and FF all make use of user-specified reference accompaniment phrases. They differ primarily in the way they adjust the pitches of the reference phrases to match the melody and the chords.

4. EXPERIMENTS AND DISCUSSIONS

4.1 Data Preparation

Two public datasets, namely Pop909 [21] and Wikifonia-Music-Text (WKF) [9] were used for the accompaniment generation test. Pop909 contains lead sheets with polyphonic piano accompaniments in pop style. It contains 857 songs with 11,041 phrases in 4/4 time, sufficient for training deep neural networks. In contrast, WKF contains lead sheets without accompaniment for many genres. 80 WKF pop songs with 189 phrases were chosen as test examples.

Pop909 and WKF contain songs that are organized into multiple sections. Each section contains 2 to 16 phrases. Each phrase consists of 8 to 32 fragments, in multiples of 8 fragments. Only songs with 4/4 time signature were used because the DNN method of WSG works only on songs with 4/4 time. Each fragment is half bar long. Similar to existing work, their tempo was set to 120 beats per minute and the songs were transposed to all 12 music keys to ensure that WSG could handle test melodies of various keys. Each song was encoded in step-function representation. Recall that the desired contrast between the generated accompaniments in different melody sections should depend on user's preference (Section 4.2). To eliminate subjective variability, quantitative evaluation was performed only on the main section, i.e., section A, of the songs.

The Pop909 dataset was split into training and validation sets and WKF served as the testing set. The training, validation and test set ratio was 10:1:1. Training and validation are required for AM and WSG. In contrast, FlexFlow does not require training. Audios of the test results are available in `https://tinyurl.com/flexflow-db`.

4.2 Quantitative Evaluation

(a) Test Procedure

The objective of this test is to evaluate the performance of FlexFlow (FF). Three versions of FF, which use different pitch adjustment options, are compared. FF1 applies pitch adjustment Option 1 to obtain chord pitches and FF2 uses Option 2 to minimize dissonance between accompaniment and melody. FF1.5 balances Option 1 and Option 2 to combine their advantages. In addition, AccoMontage (AM) and Whole Song Generation (WSG) were also tested.

FF and AM can use multiple user-selected reference phrases, whereas the publicly available WSG supports only single reference phrase. For fair comparison, all methods are restricted to use a single common reference accompaniment phrase for the whole melody. To emulate the selection of appropriate reference phrase by a human user, a separate AM was run on the testing melody, and the reference phrase selected by this AM for the first melody phrase was used as the common reference phrase.

(b) Quantitative Test Metrics

For objective comparison, a set of musically meaningful metrics were identified:

- Bass hit ratio B
  An accompaniment fragment g_j should be set at the right pitch level to harmonize with the melody according to its associated chord c_j. This is achieved when the fragment's lowest pitch is the same as the bass pitch of its associated chord, a condition called bass hit. Let n_b denote the number of fragments that achieve bass hit and n denote the total number of fragments in the song. Then, the bass hit ratio B measures the fraction n_b/n. A higher B value indicates better match between the accompaniment and the associated chords.

- Chord pitch ratio R0
  This metric measures the ratio of chord pitches n_c to non-chord pitches n_n in the accompaniment fragment g_j with respect to the associated chord c_j: R0 $= n_c/(n_c + n_n)$. It measures the harmony between the accompaniment and the associated chord. The higher the R0 value, the better is the match between them.

- Dissonance D
  The average dissonance D between melody and accompaniment is defined in Eq. 2. The lower the D value, the more harmonious is the match between the accompaniment and the melody.

- Contour Difference C
  Contour difference measures the consistency between the generated accompaniment phrases in a melody section. It is based on the difference in pitch contour between corresponding fragments g and g' measured in terms of their DCT vectors $\mathbf{v}$ and $\mathbf{v}'$ (Section 2.2):

$$\phi(g, g') = ||\mathbf{v} - \mathbf{v}'||. \tag{4}$$

  Contour difference is the average difference over all pairs of corresponding accompaniment fragments in the same section S:

$$\mathrm{C} = \frac{1}{|S|^2} \sum_{g,g' \in S} \phi(g, g'). \tag{5}$$

  The lower the C value, the more similar are the contours of accompaniment phrases.

- Contour Difference C$'$
  This metric is a variant of contour difference C. It excludes fragments at the end of a phrase or a section that are adjusted to ensure proper closure.

Table 1. Quantitative evaluation of accompaniment generation methods.

	Chord match		Melody match	Contour Difference		
	B↑	R0↑	D↓	C↓	C′↓	C′/C↓
AM	0.67	0.93	0.40	29.81	26.77	90%
WSG	0.43	0.60	0.46	262.68	245.31	93%
FF1	**1.00**	**0.99**	0.39	**15.22**	**9.54**	**63%**
FF1.5	1.00	0.85	0.34	35.03	27.40	78%
FF2	1.00	0.71	**0.32**	39.31	31.39	80%

(c) Quantitative Comparison of Various Methods

Table 1 tabulates the quantitative results of all the methods. All FF variants achieve the perfect bass hit ratio B of 1.0, indicating that the accompaniment fragments are set at the bass pitches of their associated chords. This is achieved by Line 5 of FF algorithm. In contrast, AM and WSG achieve lower B values as they do not handle bass hit explicitly.

In theory, FF1 should achieve the perfect chord pitch ratio R0 of 1.0 because it sets all accompaniment pitches to chord pitches. Test result indicates that FF1's R0 is very close to but not exactly 1.0. This is due to a small amount of held pitches that cross fragment boundaries. As held pitches continue unchanged across fragment boundaries, they may not be chord pitches with respect to the chords of the succeeding fragments. A possible extension to FF1 is a boundary-aware constraint that sustains pitches across fragment boundaries only if they belong to the chord of the succeeding fragment. FF2 minimizes dissonance instead of keeping chord pitches and FF1.5 balances the options of keeping chord pitches and reducing dissonance. Therefore, among the three variants, FF2 achieves the lowest R0. WSG achieves the lowest R0 among all methods because it can only handle chord pitches as soft constraints. All methods achieve a low average dissonance D of 0.46 or less. This is due to the methods' matching of accompaniment phrases to melody phrases. FF2 achieves the lowest D value because it explicitly minimizes dissonance between the accompaniment and melody fragments.

Test results about contour difference C indicate that FF1 best retains the contour of the reference accompaniment phrase in the generated accompaniment phrases. In other words, FF1 best maintains consistency among generated accompaniment phrases, which is important for preserving the style of reference accompaniment. By reducing dissonance, FF1.5 and FF2 cause more changes to phrase contours, resulting in larger C values. AM retains the reference contour well, and achieves the second lowest C value.

In contrast, WSG does not impose consistency in the generated accompaniment phrases, and has very high C value.

To handle closure, FF variants adjust the rhythmic pattern of the accompaniment fragments at the end of phrase and section with the structure label of 'C'. This adjustment is expected to change the phrase contours and results in higher contour difference. By discounting closing fragments, C' values give better indication of how well the methods maintain consistency in the generated accompaniment. In direct correlation to the C values, FF1 has the lowest C' value, whereas WSG has the highest C' value. AM, FF1.5 and FF2 have roughly the same lower C' values than does WSG.

The ratio C'/C indicates the amount of phrase and section closure handled by a method. FF1 explicitly handles closure and achieves the lowest C'/C ratio of 63%. FF1.5 and FF2 also explicitly handle closure. Nevertheless, their effects are less apparent due to their larger C values. In comparison to FF1, they achieve a higher C'/C ratio of about 80%. AM and WSG have very high C'/C ratios of 90% or more, showing that they do not handle closure.

In summary, FF1 has the best overall performance, with AM and FF1.5 following closely behind FF1. FF variants achieve good performance because the accompaniment generation algorithm can perform fine-grained adjustment of individual pitches in the generated accompaniment. AM applies VAE to adjust pitches to match melody and chords. WSG uses diffusion models to perform overall pitch adjustment, and lacks fine-grained control for simultaneously handling multiple conditions of pitch progression, accompaniment-chord matching, accompaniment-melody matching and closure.

4.3 Subjective Evaluation

A user study was conducted to compare the perceived accompaniment quality of various methods. 40 participants aged 18 to 65 years (average age = 41.6 years, SD = 11.8 years) were recruited via CloudResearch. All participants reported having at least five years of music training and being able to play the piano. Each participant was presented with two 8-bar songs with polyphonic accompaniment. One of the pair was either the human-composed (HC) accompaniment or the results of FlexFlow with Option 1. The other was produced by AM or WSG. The participant was asked to pick the more harmonious result.

Note that there is no ground truth in accompaniment generation as there are more than one harmonically pleasing accompaniment for a given melody. The HC accompaniments do not serve as ground truth but are used to verify whether the participants' assessment is reliable. For example, if the participants judge a HC accompaniment to be less harmonious than a DNN's result, it does not mean that the HC accompaniment is actually less harmonious than the DNNs' result. Instead, it suggests that the participants' judgments may be influenced by factors other than harmony quality, making the assessment unreliable. On the other hand, if the particpants judge HC accompaniment to be more harmonious than a DNN's result, it means that their assessment is reliable. Therefore, FlexFlow is not compared with HC accompaniment directly, which also helps to reduce listener's fatigue.

Table 2. User study results for accompaniment generation.

Cases	AM	WSG
H+	61 (49.2%)	87 (70.7%)
H−	63 (50.8%)	36 (29.3%)
F+	70 (56.9%)	84 (68.3%)
F−	53 (43.1%)	39 (31.7%)
H+ and F+	38 (31.2%)	67 (55.4%)
H+ and F−	23 (18.9%)	20 (16.5%)
H− and F+	31 (25.4%)	16 (13.2%)
H− and F−	30 (24.6%)	18 (14.9%)
Total votes	122 (100%)	121 (100%)

Each participant repeated the above procedure for 20 pairs of songs. The participants' assessments were grouped into one of the following cases:

- H+: Human-composed accompaniment is more harmonious than DNN's result.
- H−: Human-composed accompaniment is less harmonious than DNN's result.
- F+: FlexFlow's result is more harmonious than DNN's result.
- F−: FlexFlow's result is less harmonious than DNN's result.

Table 2 indicates that a large majority of participants judged HC accompaniments to be more harmonious than WSG's results, and roughly equal number of them judged either HC accompaniments or AM's results as more harmonious than the other. These results show that the participants understand and appreciate accompaniment. Therefore, the user study results are reliable. The majority of participants judged FF's results to be more harmonious than those of AM and WSG (56.9% and 68.3% respectively). Moreover, the majority of participants judged both FF's results and HC accompaniments as more harmonious than those of AM and WSG (H+ and F+ case). These results show that the participants judged FlexFlow's accompaniments to be more harmonious than those of AM and WSG, and its quality is closer to human-composed accompaniments compared to that of DNNs.

4.4 Qualitative Evaluation of Various Methods

For this test, AM, WSG and FF were applied to the song *Sentimental Me* with the section structure of A8, A8, B8, A8, where Section B contrasts with Section A. Each section consists of 8 bars. Section A and Section B should have contrasting reference accompaniment phrases. To emulate the selection of reference phrases by a user, a separate AM was run to select the reference phrases from a reference set. Then, the first reference phrases selected for the first Section A and Section B were used as the common reference phrases for the test. Both reference phrases are 8 bars long. Figure 1 illustrates that the accompaniment style of Section B differs from that of Section A. Both FF1 and AM can reproduce the contrasting styles of the reference phrases between Section A and Section B. However,

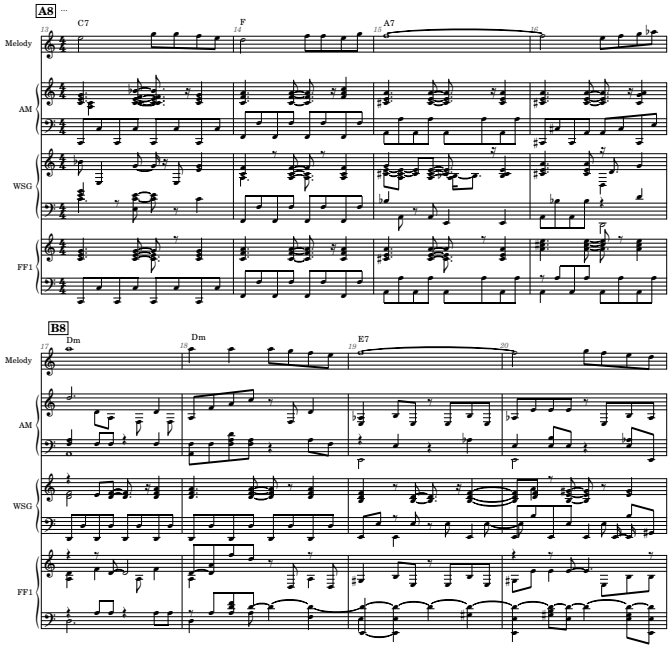

Figure 1. Accompaniment generated by various methods for *Sentimental Me*. Bars 13–16 belong to the second Section A and bars 17–20 belong to Section B.

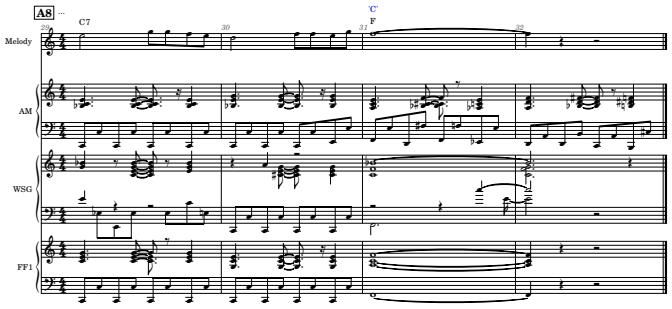

Figure 2. Closing accompaniment generated by various methods for *Sentimental Me*.

their treatments of harmonic contrast differ. The bass pitch of FF1's accompaniment progresses from A to D, which conforms to a perfect cadence and enhances the harmonic distinction between Section A and Section B. FF1 achieves this by ensuring bass hit, i.e., the lowest-pitch of accompaniment fragment is the same as the bass pitch of the associated chord. On the other hand, AM produces a bass pitch A that is sustained from bar 16 to bar 17, as the chord transits from A major to D minor. As pitch A appears in both A major chord (A–C♯–E) and D minor chord (D–F–A), it reduces the perceptual salience of harmonic contrast. The publicly available WSG does not support specification of different reference phrases for different sections. Thus, it cannot produce section contrast.

Figure 2 illustrates the last four bars of the song. FF1 produces a F major chord held from bar 31 to bar 32, with the same duration as the melody note, resulting in a clear closure to the song. In contrast, AM continues the regular rhythmic pattern, making the accompaniment sound open-ended. WSG produces a held chord from bar 31 to bar 32. In addition, it adds a short sixteenth-note chord at the end of bar 31. This creates a slight sense of tension before the song ends, disrupting closure. Moreover, pitches in the last poly-pitch (C–F–G–B♭) do not match the F major chord

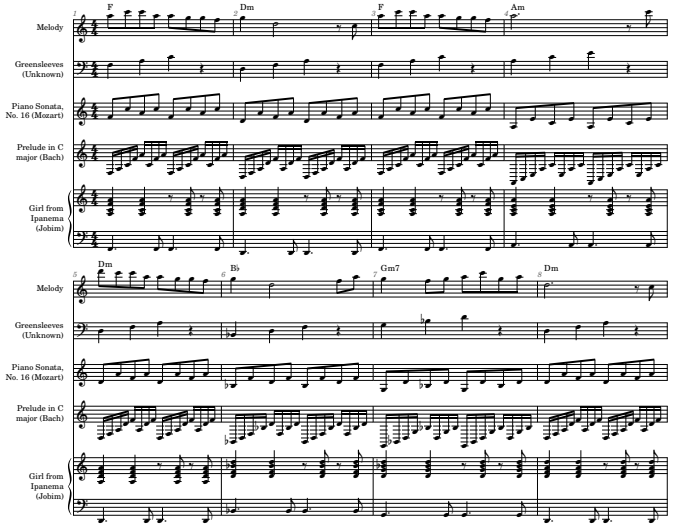

Figure 3. Accompaniments generated by FF across styles.

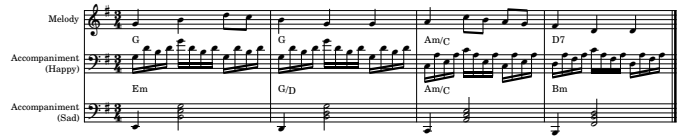

Figure 4. Happy and sad sounding accompaniments by FF.

exactly. This undermines harmonic coherence.

In summary, FF1 maintains section consistency while allowing for contrast in accompaniment across distinct sections. It ensures section closure at the end of section. AM also preserves coherence within a section and allows for contrast across distinct sections. However, it does not ensure proper section closure. WSG cannot cater to section coherence, contrast and section closure.

4.5 Accompaniment in Various Styles

Figure 3 illustrates FlexFlow's ability to generate accompaniment based on different references that reflect the folk, classical and jazz genres. The reference phrases are taken from *Greensleeves* (Row 2), Mozart's *Sonata No. 16* (Row 3), Bach's *Prelude in C* (Row 4) and *Girl from Ipanema* (Row 5). By generating accompaniment that matches the specific reference phrase, FF1 consistently generates accompaniment that matches the reference style. For example, Fig. 3 row 2 illustrates a slow folk style, row 3 illustrates a slightly faster classical style, and row 4 illustrates a fast-paced baroque style. Row 5 shows FF1's flexibility in adapting to jazz idioms, producing a bossa nova-style accompaniment. Fig. 4 further demonstrates FF1's flexibility in generating happy and sad sounding accompaniments for the same melody by varying target chords and reference accompaniments. Moreover, the melody has a time signature of 3/4 instead of the usual 4/4, further demonstrating FlexFlow's flexibility is adapting to various melody styles.

5. FUTURE WORK AND CONCLUSION

The proposed method FlexFlow applies constrained discrete optimization to generate style-specific accompaniment without the need to train on large datasets. It adjusts

individual pitches according to target chords and reduces dissonance. It preserves pitch directions and pitch differences to achieve style-specificity. In contrast, comparable DNN methods perform approximation by learning to generate outputs that are as close as possible to the training outputs. They can only handle soft constraints in the form of weighted sum of loss function. Thus, they are not as precise as FlexFlow in handling pitches and dissonance. Note that two closest pitches have the largest dissonance. A small error in DNNs' pitch generation can create large dissonance between the accompaniment and the melody. Therefore, DNNs' results are less harmonious than FF's results, as judged in the user study.

Quantitative and qualitative evaluations show that FlexFlow can generate accompaniment in the style specific to user-provided reference phrases, with as little as one reference phrase. It can handle full-length melodies with multiple sections. It is able to achieve section coherence and produce contrast between sections.

At present, FlexFlow retains the rhythm of a single reference phrase throughout each section. A future extension of FlexFlow is to introduce rhythmic adaptation that would allow the accompaniment to vary its pacing or rhythmic density based on the melody, while maintaining stylistic consistency with the reference phrases. Another extension is to introduce explicit user control over aspects such as rhythmic density, voice thickness or stylistic emphasis, while matching the melody structure.

6. REFERENCES

[1] D. Cope, *Computers and Musical Style*. Madison, Wisconsin: A-R Editions, 1991.

[2] M. T. Pearce, "Statistical learning and probabilistic prediction in music cognition: mechanisms of stylistic enculturation," *Ann N Y Acad Sci.*, vol. 1423, no. 1, pp. 378–395, 2018.

[3] Y. Mo, "Designing an automatic piano accompaniment system using artificial intelligence and sound pattern database," *Mobile Information Systems*, 2022.

[4] H.-M. Liu, M.-H. Wu, and Y.-H. Yang, "Lead sheet generation and arrangement via a hybrid generative model," in *Proc. of the Int. Society for Music Information Retrieval Conf. (ISMIR)*, 2018, pp. 23–27.

[5] H. Liu and Y. Yang, "Lead sheet generation and arrangement by conditional generative adversarial network," in *Int. Conf. on Machine Learning and Applications (ICMLA)*. IEEE, 2018, pp. 722–727.

[6] H. Zhu, Q. Liu, N. J. Yuan, K. Zhang, G. Zhou, and E. Chen, "Pop music generation: From melody to multi-style arrangement," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 14, no. 5, pp. 1–31, 2020.

[7] W. Hsiao, J. Liu, Y. Yeh, and Y. Yang, "Compound word transformer: Learning to compose full-song music over dynamic directed hypergraphs," in *AAAI*, 2021, pp. 178–186.

[8] H. Zhu, Q. Liu, N. J. Yuan, C. Qin, J. Li, K. Zhang, G. Zhou, F. Wei, Y. Xu, and E. Chen, "Xiaoice band: A melody and arrangement generation framework for pop music," in *ACM SIGKDD Conf. on Knowledge Discovery & Data Mining*, 2018, pp. 2837–2846.

[9] Z. Wang, L. Min, and G. Xia, "Whole-song hierarchical generation of symbolic music using cascaded diffusion models," in *Int. Conf. on Learning Representations (ICLR)*, 2024.

[10] J. Zhao, G. Xia, Z. Wang, and Y. Wang, "Structured multi-track accompaniment arrangement via style prior modelling," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 37, pp. 102 197–102 228, 2024.

[11] E. Cambouropoulos, "A general pitch interval representation: Theory and applications," *Journal of New Music Research*, vol. 25, no. 3, pp. 231–251, 1996.

[12] B. Cornelissen, W. H. Zuidema, J. A. Burgoyne *et al.*, "Cosine contours: a multipurpose representation for melodies." in *Proc. of Int. Society for Music Information Retrieval Conf. (ISMIR)*, 2021, pp. 135–142.

[13] D. Fox and D. Weissman, *Chord Progressions: Theory and Practice: Everything You Need to Create and Use Chords in Every Key*. Alfred Music, 2013.

[14] R. Smith, D. Anderson, C. Payne, and T. Herson, *Hooktheory I: Music Theory. Book 1*. Hooktheory LLC, 2018.

[15] P. Harrison and M. T. Pearce, "Simultaneous consonance in music perception and composition." *Psychological review*, vol. 127, no. 2, p. 216, 2020.

[16] D. Huron, "Interval-class content in equally tempered pitch-class sets: Common scales exhibit optimum tonal consonance," *Music Perception*, vol. 11, no. 3, pp. 289–305, 1994.

[17] S. Y. H. Lew, W. K. Leow, and K. R. Agres, "Style-specific melody harmonization via multi-objective optimization," *Int. Journal of Music Science, Technology and Art*, vol. 7, no. 2, pp. 58–73, Jul. 2025.

[18] D. Temperley, "The cadential iv in rock," *Music Theory Online*, vol. 17, 04 2011.

[19] G. Hadjeres, F. Pachet, and F. Nielsen, "DeepBach: a steerable model for Bach chorales generation," in *Int. Conf. on Machine Learning (ICML)*, 2017, pp. 1362–1371.

[20] Z. Yin, F. Reuben, S. Stepney, and T. Collins, "Deep learning's shallow gains: a comparative evaluation of algorithms for automatic music generation," *Machine Learning*, vol. 112, no. 5, pp. 1785–1822, 2023.

[21] Z. Wang, K. Chen, J. Jiang, Y. Zhang, M. Xu, S. Dai, G. Bin, and G. Xia, "Pop909: A pop-song dataset for music arrangement generation," in *Proc. of the Int. Society for Music Information Retrieval Conf. (ISMIR)*, 2020.