

Response to Reviewers' Comments

Reviewer 1:

Point 1.1 The literature survey seems to be sufficient. However, there are several references with incomplete information, such as [GD97], [HPN17]. The candidate should do a thorough proofreading to rectify such problems.

Reply & Changes: This has been fixed in the Bibliography section.

Point 1.2 The candidate has given sufficient technical details for reproducibility. But Algorithm 1 in page 37 of the thesis proposal is a bit confusing. It has not clearly illustrated how style-specific harmonization and accompaniment is implemented in the algorithm.

Reply & Changes: Style is described by the chord transition model derived from one or more reference music pieces (Section 4.2.2 Pg. 34). It is also clarified in Section 4.2.5 (Pg. 35-36) and the input of the algorithm.

In addition, a new Section 1.3 Regarding Style (Pg. 3-4) has been added to describe music style that can be interpreted at different granularity.

Reviewer 2:

Point 2.1 Section 1.3 is unnecessary and should be omitted. The Table of Contents says the same thing. Instead, please give a list of contributions of your work. Contributions is not what you have done, but the BENEFITS of your method over existing works. Example, your method is faster, or more general than existing methods.

Reply & Changes:

A new Section 1.4 (Pg. 4) is added to highlight the contributions of thesis work. We choose to retain the Section 1.5 on Thesis Proposal Organization because it describes the relationships between the sections.

Point 2.2 Sec 3.3 is odd. It shouldn't be placed after the chapter summary.

Reply & Changes: Section 3.1.6 and 3.2.6 (Pg. 19-20, 26-29) are not summaries of Chapter 3. Instead, they are summaries of Section 3.1 and Section 3.2 which describe related works on harmonization methods and accompaniment methods.

Section 3.3 (Pg. 30) is a section at the same level as Section 3.1 and 3.2. It discusses studies of deep neural networks' performance on music generation, not harmonization and accompaniment generation. It is added to provide a broader perspective into the performance of deep neural networks in comparison to Markov Models.

Point 2.3 On page 35, you formulated your problem as a constraint optimization: three objectives and one constraint. Yes, your FlexChord algorithm is *not* the usual

optimization. Please explain why you didn't solve it as a multi-objective constraint optimization.

Page 35 last para: give a reference for the claim that the lexicographic method produces all Pareto optimal solutions.

Reply: Section 4.3.2 (Pg. 37-39) states that FlexChord indeed performs multi-objective optimization by adopting the modified lexicographic method. The topic of multi-objective optimization is always framed as a constrained optimization problem, so the word "constrained" is customarily dropped from the topic title of "Multi-objective optimization".

Section 4.3.1 (Pg. 36-37) clarifies that the lexicographic method does not produce all Pareto optimal solutions. It also explains why the modified lexicographic method can produce all Pareto optimal solutions. A reference to this explanation is also added.

Point 2.4 In your FlexChord algo, carefully explain where the "example style" is provided as input. How is the reference chord set C created from an example music, eg. from Psycho's spooky motif? Is this C manually created? Can it be automated? How?

Reply: The music style is captured in the chord transition model C (Section 4.2.2, Pg. 34). It is created by measuring the frequency of occurrence of chord pairs in the example music. The creation of C is automated in data preparation stage (Section 4.4.1, Pg. 40).

Point 2.5 This means that, your complete method has 2 steps:

A: Given an example music, E (which contains the style), determine the chord set C.
B: Input C and the target melody M to the FlexChord algo.

Thus, you need to compare the performance of A+B with any existing method X, and not just compare B with X.

As this is a CS thesis, give the runtime complexity of A and B. Big-O notation preferred.

Reply: Part A is outside of the algorithm. It is the data preparation stage and not part of the method, just like how training examples of the deep neural networks are prepared.

A new section 4.3.3 (Pg. 39-40) is added to describe the time complexity of FlexChord.

Point 2.6 In your experiments, give some examples where your method failed, and suggest (no need to prove) why it failed.

Reply: As discussed in Section 4.3.2 (Pg. 37-39), the algorithm satisfies all the optimization objectives and constraint. Therefore, it always produces a valid solution. It is not brittle and does not fail. A possible way for the algorithm to produce incomplete result is when the chord transition model does not contain all the

necessary chord pairs. But this situation will never happen because chord pairs that do not appear in example music pieces are still present in the chord transition model with a transition probability of zero.

Point 2.7 It is imperative to conduct a human perception test, even though this is tedious and difficult. Please work with your co-supervisor to do this. Even a small set of human listeners will greatly boost the credibility of your work.

Reply & Changes: A human perception test has been conducted. The test results indicate that FlexChord's harmonizations are better than those of the DNNs, and are comparable to those of the original harmonizations. Details of the test are reported in Section 4.4.4 (Pg. 49-50).

Point 2.8 You should play audio examples EARLY in your presentation, and not wait until towards the end. This will also better explain the problem you are trying to solve. Ironical that for a work on music, nothing was heard until close to the end.

Reply: Noted. For subsequent music research presentation, I will play the 1st audio example when introducing the music problem.

Point 2.9 Also, the various metrics need to be briefly explained: what are the possible range of values? Is a larger value better, or smaller? Is there an ideal value?

Reply & Changes: The various metrics are explained in part (b) of Section 4.4.2 (Pg. 43-44). Moreover, the arrow signs in Table 4.3 (Pg. 45) indicate whether the preferred values of the metric should be high or low. By definition, all metrics are positive. The ideal values of the metrics are unknown because music harmonization has no ground truth. They depend on the style of the music.

STYLE-SPECIFIC HARMONIZATION AND ACCOMPANIMENT

by

STEPHANIE LEW

**A THESIS PROPOSAL FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY**

in

COMPUTER SCIENCE

in the

SCHOOL OF COMPUTING

of the

NATIONAL UNIVERSITY OF SINGAPORE

NOVEMBER 2024

Supervisors:

Associate Professor LEOW Wee Kheng, Main Supervisor
Assistant Professor Kathleen AGRES, YSTCM, Co-Supervisor

Abstract

Automated music composition (AMC) aims at applying computer algorithms to create music. It enables musical novices with limited knowledge to create new music and helps professional musicians to enhance their creative process. More importantly, it fosters collaboration between musicians and non-musicians, leading to diverse applications like music for videos, games, and therapy.

AMC is not a single, monolithic task. In the broadest sense, AMC can be divided into two main tasks: (1) generation of music structure and melody and (2) generation of chords and accompaniment. The first task of generating melody is well-studied. The second task is important for generating music with multiple voices that also harmonize with each other. Therefore, this thesis focuses on the task of generating chords and accompaniment given the melody of a target music piece, complete with its music structure.

A variety of methods have been developed to generate chords and accompaniment, but they have some shortcomings. Most existing methods are trained on a large training set of a specific genre such as pop or jazz. To accommodate to different styles, they need to train a different version of the method on a different style-specific training set, increasing the total training time. If they are trained on a mixed-style set that contains examples of various styles, they will mix up the styles and produce results that are diluted in style. Moreover, existing methods cannot cater to cadence and voice leading. Deep neural networks (DNNs), in particular, only work on 8-bar phrases instead of complete music pieces. Thus, they cannot handle global music structure.

This thesis proposes a method of harmonization called FlexChord that overcomes the shortcomings of existing methods. FlexChord achieves style specificity by producing chord progressions that match the style of a selected set of example music, which can be as little as a single music piece. It applies non-iterative, multi-objective optimization during chord generation that does not require training. Thus, it can easily produce chord sequence of a specific style or a custom style. The requirement of cleaning, annotating and training on large datasets is avoided. In addition, FlexChord can handle cadence and voice leading effectively. It can generate chords that match full-length melodies, complete with global music structure.

FlexChord shows promising results as a flexible harmonizer that can create pop and jazz chord progressions for complete songs. Human perception test results show that FlexChord’s harmonization is better than those of comparable DNNs and is close to that of human harmonization. The chords generated by FlexChord can be utilized for generating style-specific accompaniments to full-length melodies, which is the remaining work of this thesis. Generation of accompaniment involves transforming the generated chords into multiple rhythmic voices that fit a specific accompaniment style. The generated accompaniment should also match the global music structure of the given melody.

Contents

Abstract	i
List of Figures	v
List of Tables	vi
1 Introduction	1
1.1 Motivation	1
1.2 Research Objectives	2
1.3 Regarding Music Style	3
1.4 Contributions of Research	4
1.5 Organization of Thesis Proposal	4
2 Music Background	5
2.1 Basic Elements of Music	5
2.2 Formal Structure of Music	8
2.3 Principles of Harmonization	10
2.4 Principles of Accompaniment	12
3 Related Work	13
3.1 Harmonization	13
3.1.1 Rule-based and Grammar-based Methods	13
3.1.2 Genetic Algorithms	14
3.1.3 Deep Neural Networks	15
3.1.4 Markov Models	17
3.1.5 Direct Optimization	18
3.1.6 Summary	19

3.2	Accompaniment Generation	22
3.2.1	Rule-based and Grammar-based Methods	22
3.2.2	Genetic Algorithms	23
3.2.3	Deep Neural Networks	24
3.2.4	Markov Models	25
3.2.5	Hybrid Methods	26
3.2.6	Summary	26
3.3	Performance of Deep Neural Networks for Music Generation	30
4	Harmonization	31
4.1	Representation of Music Elements	31
4.2	Problem Formulation	33
4.2.1	Chord-Melody Matching	33
4.2.2	Chord Progression and Music Style	34
4.2.3	Cadence	34
4.2.4	Voice Leading	35
4.2.5	Optimization Objectives and Constraint	35
4.3	Multi-Objective Optimization	36
4.3.1	Overview	36
4.3.2	FlexChord Harmonization	37
4.3.3	Analysis of Time Complexity	39
4.4	Experiments and Discussions	40
4.4.1	Data Preparation	40
4.4.2	Quantitative Evaluation	42
4.4.3	Qualitative Evaluation	47
4.4.4	Subject Evaluation	49
4.4.5	Flexible Harmonization Style	50
4.4.6	Full-Length Test	53
4.4.7	Summary	55
5	Remaining Work and Conclusion	57
	References	59

List of Figures

2.1	Elements of a music piece.	6
2.2	Melody as a single voice.	6
2.3	Melody with 1 supporting voice.	6
2.4	Melody with accompaniment of 4 voices.	7
2.5	Lead sheet containing melody and chord labels.	7
2.6	Formal music structure of Bach's <i>Minuet in G</i>	9
2.7	Collapsed music form of <i>Twinkle, Twinkle, Little Star</i>	9
4.1	Step function of a melody.	32
4.2	Chords generated by various methods for the first 8 bars of <i>Colors of the Wind</i>	48
4.3	The original and jazz chords generated by FC for bars 17 to 24 of <i>Israel</i>	51
4.4	Pop and jazz chords generated by FC for the first 8 bars of <i>Colors of the Wind</i>	52
4.5	Spooky chords generated by FC for the first 8 bars of <i>Hall of the Mountain King</i>	53
4.6	Selected fragments of test results for the full song of <i>Colors of the Wind</i>	54

List of Tables

3.1	Comparison of methods for harmonization.	20
3.2	Comparison of methods for accompaniment generation.	29
4.1	Pitch dissonance depends on pitch interval [Hur94].	33
4.2	Summary statistics of the datasets used for a comparative study.	41
4.3	Quantitative evaluation for chord diversity and chord-melody match.	45
4.4	Subject evaluation results.	50

Chapter 1

Introduction

1.1 Motivation

Automated music generation (AMG) or automated music composition (AMC) aims at applying computer algorithms to create music. It enables musical novices with limited knowledge to create new music and helps professional musicians to spark new creative ideas by generating a wide range of musical ideas quickly, speeding up the creative process. More importantly, it can facilitate new forms of collaboration between musicians and non-musicians resulting in a variety of applications such as music for videos, games and clinical therapy [HS13].

A variety of methods have been developed to generate music compositions automatically in the past decade. For example, rules and grammars [HS13; CLC13; Agu+10; WC16b], and Markov models [CC11b; BC15; Mo22] are used to generate accompaniments, whereas integer programming is used to incorporate constraints for the accompaniment in the music structure [CSH18]. Genetic algorithms are used to generate variations of melody and accompaniment [GJ02; LT12; KGR18]. More recently, deep neural networks are trained to generate music in the style of Bach [Lia16; HPN17; Hua+17], pop [Zhu+18; Zhu+20; Che+21; Rhy+22; Yi+22] and folk music [ZX21]. More details of automated music generation are available in [HCC17; BHP20; Civ+22].

Despite the success of recent deep neural networks (DNNs) for AMC, they still have shortcomings. For example, they are restricted to training and testing on 8-bar phrases rather than complete music pieces [BP20]. Thus, they do not handle global structure of complete music pieces. They lack meaningful controls for a composer to constrain the music according to a specific style [Dég+22]. Furthermore,

their generated music lacks novelty. According to performance studies on deep neural networks for music [Yin+21; Yin+23], the state-of-the-art DNNs tend to produce copies of training examples and their performance improvement is negligible compared to Markov methods for music generation.

In addition, existing methods typically handle music style at the coarse level of genre such as pop and jazz. They can generate music for different genres but not necessarily music that conveys the styles of various composers and emotions. Most existing methods are trained on a training set of a specific style. Training on a mixed-style set that contains examples of different styles causes the methods to mix up various styles and produce results that are diluted in style. To accommodate to different styles, they need to train a different version of the method on the training examples of a different style, increasing the total training time needed. Notably, existing deep neural network methods require human preparation of training sets and application of iterative optimization during training, which are tedious and time-consuming. Moreover, they cannot cater to cadence and voice leading, and only work on 8-bar phrases rather than full-length music pieces. Thus, they cannot handle global music structure. This proposed thesis research aims to address these shortcomings.

1.2 Research Objectives

The overall research goal is to develop an AMC framework. AMC is not a single, monolithic task. It consists of several functional components that aim at the four essential elements of music, namely melody, harmony, rhythm and timbre [HCC17]. In the broadest sense, AMC can be divided into two main tasks: (1) generation of music structure and melody and (2) generation of chords and accompaniment. The problems of generating music structure and melody are considered together as a single task as music structure relates to phrase repetition and contrast in the melody. The combination of phrase repetition and contrasts lead to sections and overall music form in complete music. The second task is the generation of chords and accompaniment that support a melody. The first task of generating melody is well-studied [Jaq+16; JKW17; Lop+17; CSG17; Wil+17; JKA17; Goi+19; HH20; Hua+20; SGA20; HBE20; SW20; GDG21; TT21; Sab+21; Sut+21; Dai+21]. Therefore, this thesis

focuses on the second task of generating chords and accompaniment given the melody of a target music piece, complete with its music structure. The relationship between chords and accompaniment is discussed in Section 2.1.

The chords and accompaniment along with the melody constitute music with multiple voices. Another related task in the generation of music with multiple voices is chorale generation. In chorale generation, chords are expanded into individual notes for each of the four standard voices (soprano, alto, tenor and bass) in traditional chorales [Ebc88b]. It is a specific type of accompaniment generation that has been extensively studied [TA99; HFM91; PAW+99; AW04; GM07; YG07]. Moreover, well established rules in traditional chorale music can be easily enforced on each individual accompaniment voice to produce chorale music. Hence, chorale generation is regarded as a simple case of accompaniment generation and is excluded from the scope of this work.

Other important aspects to consider for AMC are instrumentation and rearrangement. Instrumentation is the assignment of instruments to voices or parts in a composition, while rearrangement is the reassignment of notes in the voices according to the pitch ranges of the instruments. They are important as the selection of instruments for rendering musical parts influences the overall timbre and sound quality of the music that is played. In this thesis, piano is chosen as the target instrument for a multi-voice composition as it offers a wide range of 88 pitches that achieves rich sound quality. Other forms of instrumentation such as string quartet and modern band with unpitched drums may be generalized from piano accompaniment, and are excluded from this work.

1.3 Regarding Music Style

In music composition, style is characterized by various elements such as melody, rhythm, harmony, voice leading, and accompaniment. The concept of style can be understood at multiple levels. At the coarsest level, style refers to the genre of music such as pop, jazz and classical. Style can also refer to music in specific historical periods such as Baroque, Classical, Romantic, etc.

At a fine level, style can refer to the composition style of individual composers. For example, Beethoven, Mozart and Haydn belong to the Classical period but

their composition styles are very different. For a prolific composer, the style of different compositions can differ significantly. For example, Chopin’s Mazurkas and Nocturnes have very different styles. At the finest level, each piece of music can have a different style. For example, Chopin’s Mazurkas No. 1 and 2 have different styles even though they are both Mazurkas [Col+11].

1.4 Contributions of Research

This research work aims to achieve style-specific harmonization and accompaniment generation. Its contributions include the following:

- Generate harmonization and accompaniment of required style at various style levels.
- Achieve style-specificity using as little as a single example music piece.
- Generate harmonization and accompaniment for custom style.
- Apply multi-objective optimization that does not require training, thus eliminating the need to clean and annotate large amounts of training data.
- Handle cadence and voice-leading that are absent in existing work.
- Generate harmonization and accompaniment for full-length melodies.

1.5 Organization of Thesis Proposal

To understand how chords and accompaniment support melody, it is necessary to first understand the elements of music and structure (Chapter 2). Music elements such as melody, chords, voices and rhythm are involved in harmonization and accompaniment generation. The principles that guide quality harmonization and accompaniment are discussed in Section 2.3 and 2.4. Related methods for harmonization and accompaniment generation are reviewed in Chapter 3. Chapter 4 discusses the preliminary work of generating harmonization. Finally, Chapter 5 concludes the thesis proposal and highlights the remaining work of accompaniment generation for completing the thesis.

Chapter 2

Music Background

2.1 Basic Elements of Music

The melody and accompaniment of a music piece comprise several elements. The most basic element in a melody is a **note**, which has three main attributes: pitch, duration and onset. **Pitch** is the perceptual property that relates to the frequency of sound and **duration** informs the length of time that a pitch is played. **Onset** is the starting time of a note. When no note is present, a **rest** is indicated. Like notes, rests have varying durations.

In conventional Western musical notation (Fig. 2.1), a sequence of notes in a melody can be divided into segments called **bars**. A bar's relative duration is encoded by the **time signature**, which specifies how many beats are contained in each bar, and which note value is equivalent to a beat. The actual duration of a bar is encoded by the **tempo**, which indicates the number of beats per minute. Melody is usually perceived by the listener as a distinct entity because of its rhythm and unique sequence of pitches. **Rhythm** refers to the pattern of sounds and silences that occur over time.

Two or more notes played concurrently form a **chord**. The notes in a chord can be played sequentially in rising and descending order. They can also be reordered in different **chord inversions**. For example, a C major chord consisting of C, E and G notes can take the **root position** (C-E-G), 2nd inversion (E-G-C) or 3rd inversion (G-C-E). These chord positions are labeled as C, C/E and C/G respectively. In addition, the notes in a chord can be packed tightly in closed position (e.g., C4-E4-G4 in octave 4) or span more than an octave in open position (e.g., C3-G3-E4 across octave 3 and 4).

Figure 2.1: Elements of a music piece. They include information about pitch, rhythm, tempo of the melody and the chords. This is an excerpt from *Gymnopédie No. 1* by Erik Satie.

Figure 2.2: Melody as a single voice. Figure 2.3: Melody with 1 supporting voice.

A chord can also be represented by a chord symbol in Roman numeral which encodes chord types with respect to a music key. For example, I, II, III are major chords, i, ii, iii are minor chords, I^+, II^+, III^+ are augmented chords and $I^\circ, II^\circ, III^\circ$ are diminished chords. Note that chord symbols do not indicate the actual position of a chord. Depending on the arrangement of its pitches, a chord can be in root position or an inversion.

The notes of a chord sounding together forms **harmony** and a sequence of chords in motion forms a **chord progression** which functions as chordal accompaniment to the melody. A chord progression of at least 2 chords that ends a phrase or section (Section 2.2) of a piece of music is called cadence. **Cadences** function like punctuation at the end of phrase, giving music phrases a sense of partial



Figure 2.4: Melody with accompaniment of 4 voices. The accompaniment includes bass and 3 inner voices.

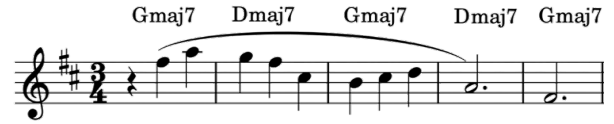


Figure 2.5: Lead sheet containing melody and chord labels.

or full resolution. When chords are present or assigned, notes in the melody or accompaniment can also be referred to as **chord tones**, if they share the same pitch name as a chord pitch, or **non-chord tones**, otherwise. Non-chord tones that transition between chord tones are called **passing tones**.

Voice refers to a sequence of notes. For example, the melody itself is a voice (Fig. 2.2). Music that consists of melody alone is monophonic music. Some music consists of melody and accompaniment that each appears as a single voice (Fig. 2.3). Accompaniment can also consist of multiple voices. For example, the music in Figure 2.4 consists of a melody, bass voice and inner voices that comprises a sequence of triad chords. This example effectively has 5 voices. A lead sheet is different from an accompaniment. In a lead sheet, the chords are located at the beginning of the bars (Fig 2.5). In an accompaniment, the chords can appear anywhere that is appropriate.

Voice leading refers to the smooth transition of notes within a voice. According to the pitch proximity principle [Hur01; Hur16], transitions that minimize voice fluctuation is considered as efficient or good voice leading. Voice fluctuation is the average difference between the pitches of successive notes in a voice.

Another important feature of music is **music texture**, which is characterized by voice density and the interactions between voices [Got+21]. Voice density is the average number of notes in a specific time span such as a bar or a beat. Voice interactions may include interactions and dependence between two voices.

2.2 Formal Structure of Music

Music can be organized hierarchically in the **form** of sections, phrases and bars. Together, these musical entities make up the formal structure of music with 3 distinct levels. Figure 2.6 illustrates a music piece that contains two sections. Each section contains 4 phrases that span 4 bars each. For sufficiently long music, a phrase may have more than 4 bars and may be organized into sub-phrases. In shorter music, such as *Twinkle Twinkle Little Star* (Fig. 2.7), a section may contain only one phrase.

A phrase may be followed by a repeat or a variation of itself, or a contrasting phrase. A phrase variation resembles the original phrase, with some minor changes to pitch, rhythm or chord. On the other hand, a contrasting phrase is melodically or harmonically distinct from the original phrase. A combination of repeating, varying or contrasting phrases form a distinct section which is typically labeled using letters (e.g., A, B). The label includes a number if it is a variation of another section (e.g., A2). Repeating, contrasting and varying sections are arranged into a **music form**.

Typical music forms include binary (A B), ternary (A B A) and rondo (A B A C A). An example of a music piece in binary form is Bach’s *Minuet in G major* (Fig. 2.6), where the opening section is concluded with a contrasting section. On the other hand, music in ternary form such as *Twinkle, Twinkle, Little Star* (Fig. 2.7) begins with a main theme, transitions to a contrasting section and concludes with the main theme. Music in rondo form can be viewed as an extension of either ternary form or binary form, where the main section (referred to as either “section A” or “refrain”) returns throughout a music piece and is juxtaposed with contrasting sections (referred to as “section B”, “section C”, or “episodes”). An example of music in rondo form is Beethoven’s *Für Elise*, where the main theme appears three times and is separated by contrasting sections.

Existing literature often use the terms short-term structure and long-term structure to refer to music structures of different length. For consistency, this thesis proposal uses these terms as follows:

- Short-term structure: relationship between nearby pitches.
- Long-term structure: relationship between music phrases.
- Formal structure: relationship between music sections.

Section A
(bars 1-16)

Phrase 1

bar

G (key): I

Phrase 2

I – V
(HC in G)

Phrase 3

Phrase 4

V – I
(PAC in G)

Section B
(bars 17-32)

Phrase 5

Phrase 6

D (key): V
(HC in D)

V – I
(PAC in D)

Phrase 7

Phrase 8

G (key): V
(HC in G)

V – I
(PAC in G)

Figure 2.6: Formal music structure of Bach’s *Minuet in G*. It consists of two sections, namely Section A and Section B. Each section contains 4 phrases and each phrase contains 4 bars. At the phrase level, typical cadences such as Half cadence (HC) and Perfect Authentic Cadence (PAC) are found at phrase endings.

Section A

Section B

Section A

Phrase 1

Phrase 2

Phrase 3

bar 1

bar 2

Figure 2.7: Collapsed music form of *Twinkle, Twinkle, Little Star*.

2.3 Principles of Harmonization

Harmonization or melody harmonization is an important functional aspect of automated music composition [HCC17]. It is the process of creating chords to match and enhance a given melody.

The key principles of harmonization include the following:

1. **Chord-Melody matching:** Chord-melody matching is the process of finding chords that go well with a melody. The quality of chord-melody matching is mainly influenced by chord-melody pitch relationships and style-specific elements aligned with the composer's preferences. Chord-melody pitch relationships can be measured in terms of dissonance, a sense of tension caused by the intervals between chord and melody pitches. The opposite of dissonance is consonance, which contributes to a sense of resolution. Music that is highly consonant can be considered boring while prolonged dissonances can be jarring and undesirable. Good quality chord-melody matching is dependent on the composer's preferred balance between dissonance and consonance.
2. **Chord Progression, Cadence and Voice Leading:** Chord progression refers to the sequence of chords in music and cadence refers to the two chord progressions located at phrase or section endings. Voice leading refers to the smooth and logical movement of individual melodic voices within a chord progression. It ensures that each voice moves smoothly from one chord to the next, creating a pleasing and coherent harmonic flow. Cadences and stylistically appropriate voice leading in bass line contribute to good quality chord progressions.
3. **Harmonic Coherence:** Harmonic coherence refers to the consistency and logical flow of chords within a piece of music. It ensures that chord progressions and voice leading make sense within the context of the melody and the key.
4. **Style Specificity:** Different musical styles have a specific set of harmonic conventions and tendencies. Understanding and adhering to these conventions can help create harmonizations that are stylistically appropriate and effective. A melody can be harmonized with different chord progressions that reflect

different styles. A composer can choose to harmonize a melody by the chord progression that suits his preferred style.

5. Novelty: Novelty in harmonization refers to sufficiently different chords and chord progression combinations not seen or experienced by the composer. It ensures that the harmony that the composer experiences is not boring and avoids the problem of frequent copying of chord progressions from existing music.

By applying these harmonization principles, chords that match and enhances the melody can be applied to enrich the overall music composition.

In addition, harmonization enhances the music structure of composition in several ways:

1. Phrases and Sections: Harmonization helps to shape the phrases and sections within a piece. The choice and arrangement of chords can create cadences (points of harmonic resolution) that mark the end of phrases or sections.
2. Form: Harmonization also helps to enhance different sections of a musical form. For example, in a typical form like AABA, each section may have a distinct harmonic progression, with the A sections often sharing a similar harmonic framework and the B section providing contrast. The choice and progression of chords in each section contribute to the overall form and provide a sense of organization and coherence.
3. Contrast: Harmonization can enhance contrasts within music through changes in harmonic progression that introduces local contrasts between phrases.
4. Development: Harmonization can aid the development of music composition through the transformation and elaboration of harmonic material such as chords, chord progressions and voicing. The transformation of relationships between a series of chords and the music key creates a sense of tonal stability or instability. In addition, the elaboration of voicings between chords create variety and interest in music texture.

2.4 Principles of Accompaniment

Accompaniment is the auxiliary part of composition designed to support the main melody in music. It can consist of one or more voices. Depending on the number of written voices, the accompaniment can be played by a monophonic instrument like violin or polyphonic instrument like piano. The key principles of accompaniment include the following:

1. Chord-Melody matching: Chord-Melody matching refers the degree of match between the chords in the accompaniment and the melody pitches. This process is important as chords are the foundation of accompaniment. Like harmonization, the quality of chord-melody matching is mainly influenced by the pitch relationships between chords and melody and the aesthetic preferences of the composer.
2. Rhythm-Melody matching: Rhythm-melody matching refers to the degree of match between the rhythm of accompaniment and the rhythm of a melody. It determines the overall groove of the music and gives sense of momentum and drive throughout a music piece. The quality of rhythm-melody matching is subject to the composer's preferences.
3. Style Specificity: Like harmonization, style-specific conventions and tendencies in accompaniment can vary among different composers. The accompaniment for a melody is not restricted to one style. A composer can choose to support a melody by the accompaniment that suits his preferred style.
4. Novelty: Novelty in accompaniment refers to sufficiently different textural and voice leading pattern not seen or experienced by the composer. This introduces unpredictability and uncertainty from a composer's perspective and improves a composer's creative process.

Applying these principles will allow the accompaniment to serve as a supportive and rich framework that enhances the melody.

Chapter 3

Related Work

The terms “harmonization” and “accompaniment” refer to different music concepts although some existing works use them interchangeably [Hua+18]. Harmonization refers to the matching of chords and melody in terms of their pitch relationships. On the other hand, accompaniment refers to the one or more voices that support the melody via chordal harmonization, rhythm and music texture. Hence, this chapter organizes the related works of harmonization and accompaniment generation in separate sections. Section 3.1 discusses existing harmonization methods that generate chord symbols or collections of pitches as held chords. The related task of generating additional voices for chorale is omitted. Section 3.2 discusses existing methods for generating accompaniment. Most of them take melody and generated chords as inputs.

3.1 Harmonization

Harmonization methods can be grouped into five categories, namely rule-based methods, genetic algorithms, Markov models, deep neural networks and direct optimization. The first two categories consist of older methods that are also reviewed in [MKS16].

3.1.1 Rule-based and Grammar-based Methods

Methods in this category use symbolic rules and grammars to generate a sequence of chords. These rules and grammars describe the structural relationships between various music elements including chord-melody matching and chord progression. They contain music notes and chords as atomic elements. Rules can be combined to represent long-term and formal structures such as phrases and sections. The rules

and grammar are manually defined based on one or more music examples. During chord generation, they are applied to generate chords that match the melody and chord progression.

Rules and grammars have been used to generate harmony as a sequence of chord symbols in the style of Palestrina [Ebc88a; Gwe02; HC05; Gog06; GC07; YT10; HS13; Agu+10], Bach [GC07] and Schoenberg [AM09]. They are usually discrete rules [Ebc88a], but can be extended to fuzzy rules [Gwe02; YT10] and probabilistic rules [Agu+10].

The strength of rule-based and grammar-based methods is their ability to capture short-term, long-term and formal structures of the example music. In addition, by constraining the harmony generation system with encoded rules and grammar, note sequences of similar harmonies can be easily generated.

Despite their strength, rules or grammars tend to be not comprehensive enough for generating all the cadences or chord sequences for harmonization. Although the rules and grammar are well defined for certain styles such as popular music and Bach chorale, they cannot be easily generalized to other complex styles unless probability is included to make probabilistic or fuzzy rules. Moreover, it is difficult to determine what combination of probabilities can be added to the rules as they may also differ between music styles.

3.1.2 Genetic Algorithms

Methods that use genetic algorithms [PATW99; Wig+99; PAW+99; GJ02; KGR18] offer some flexibility by maintaining a population of harmonic solutions to choose from, based on a fitness function. The fitness function describes the goodness of fit for chord-melody matching and chord progression. They repeat a cycle of 3 stages, namely evaluation, selection and reproduction with variation. In the evaluation stage, candidate solutions generated from user-specified examples or a random process are evaluated using a fitness function that captures chord-melody matching and chord progression. In the selection stage, a new set of candidate solutions are probabilistically copied from the old set in proportion to its fitness. In the reproduction stage, some operators are applied to the set of candidates to generate new candidate solutions. This three-stage evolution cycle repeats until

music pieces with high fitness is obtained. For effective evolution, a large population of possible music of a wide range of varieties is needed.

Methods using genetic algorithms differ mainly in the choice of fitness function. They may include weighted sum of features related to chord-melody matching, pitch intervals between voices, chord progression between two adjacent chords [PATW99; Wig+99; PAW+99] and distance between target music and training examples [GJ02]. Another variant, known as multi-objective evolutionary algorithm, involves balancing between two fitness functions [FG11; DPZZ20], one for melody and the other for chords separately. With genetic algorithms, new variations of given music can be produced easily. This is achieved by the re-selection of music phrases from existing musical pieces for modification, thus retaining short-term and possibly long-term structure and chord progression patterns.

Genetic algorithms have some shortcomings. Firstly, there is no guarantee that the optimal solution will be found. In fact, they are not able to deterministically enable harmonic patterns to evolve into concrete formal structures with contrasting sections because there is no mechanism for constraining musical phrases into a distinct sections with appropriate cadences. Another shortcoming is the difficulty of defining an appropriate fitness measure for the overall harmonization result, which needs to be changed to suit different styles.

3.1.3 Deep Neural Networks

The third category of methods uses deep neural networks to generate a sequence of chords that fit the chord-melody mapping and chord progressions of training samples [LRL17; Sun+21; Yeh+21; Che+21; Rhy+22; Ji+23]. These neural networks encode a note or chord as a numeric value. They learn a highly complex nonlinear mapping between the inputs and corresponding training outputs. During training, the weights of the neural networks are learnt using stochastic gradient descent and backpropagation. During the generation phase, the network’s output is synthesized from the features extracted from the input and the features learned from training examples by feedforward neuron activation.

Recent families of neural networks used for chord generation include long-short-term networks (LSTMs), Transformers and variational autoencoders (VAEs). LSTMs

have cyclical links between neurons that serve as a memory structure and maintain information in memory for longer periods of time. As a result, longer dependencies between states can be captured by the neural networks. The LSTM in [ZL21] generates chords automatically for specific melody segments. The methods in [LRL17; Sun+21; Yeh+21] utilize a variant of LSTM model called bidirectional LSTM to predict a chord for each bar given a melody represented in a pitch-class histogram. The method in [Yeh+21] augments the chord information in the training set with simplified harmonic function labels from music theory, whereas the method in [Sun+21] masks parts of the chord sequence during training to steer the model to generate new chords at masked positions.

Transformer networks are encoder-decoder models that use attention layers to relate different positions of a single sequence. This approach improves the networks' awareness of long-term structure. In [Rhy+22], Transformer networks are used to map melody notes to chords which are represented as either polyphonic pitch onset events or chord symbols. In addition, [Rhy+22] uses a variant of Transformer with an additional encoder to capture the relationships between melody, chords and music key. On the other hand, the VAE in [Che+21] learns the mapping between chord emotion, chords and melody. It generates chords that match the target melody and emotion.

Using deep neural networks for chord generation can be advantageous in some settings. Firstly, they do not require much knowledge of music theory. Suitable chord progressions for a given melody are learnt from existing melody-chord pairs in the training examples. Secondly, they can sometimes generate unexpected chords outside the melody key that are still harmonious to the melody [Sun+21].

Deep neural networks have several limitations. Firstly, a large training set is required to train a deep neural network. This can cause the harmonization styles of the individual training samples to be mixed and diluted in the harmonization result. Secondly, chord variety is limited to chord types in training examples. This is evident in [Sun+21; Yeh+21] where chord types are restricted to simple triads or a few seventh chords. Therefore, their chord progressions are stylistically limited. Moreover, these networks require additional data pre-processing such as transposing melodies and chords to the C key [Rhy+22]. Trained networks can only harmonize a melody with the specific style of the training set. They require re-training or

fine-tuning to harmonize in other styles [Yi+22].

3.1.4 Markov Models

Like deep neural networks, Markov models [PEB06; Pac; Tsu+17] also learn chord-melody matching and chord progression from training samples. However, they require smaller training sets than those used for deep neural networks. They describe the statistical properties of the change of state that captures the pitch of a note or pitches of a chord. They typically assume that the current state is dependent only on the immediately preceding state. They can be extended to higher-order Markov models that describe the statistical properties of the current state in terms of more than one past states.

Markov models with observable states are called Markov chains. Since their states are observable, their state change probabilities can be calculated directly from the input data samples and represented in a state transition matrix. Markov chains are used in [CC11a; GN21] to generate chords for melody. Their state transition probabilities are calculated using frequency counts of state transitions present in the training set. In [CC11a], Markov chains are combined with music-theoretic rules to model the harmonization process. It relies on only a few training samples that capture a specific style. The chord transition probability in [GN21] is defined based on neo-Riemannian principles [FM81] of music theory and includes a control parameter for the conservativeness of chord types, defined in terms of neo-Riemannian chord distances.

Markov models with unobservable states are called hidden Markov models (HMMs). They capture both state transition probabilities and output probabilities which are typically estimated based on maximum likelihood approach via algorithms such as expectation maximization and Markov chain Monte Carlo. HMMs have been used for several melody harmonization works [LJ04; SMB08; RFV13; KP+23] to model the relationship between a hidden chord sequence and the observed melody sequence in training data. They apply the Viterbi algorithm, a kind of dynamic programming algorithm, to obtain a globally optimal chord sequence from all possible chord sequences in the training data. Also, these HMMs generate pitches for notes with fixed durations and do not generate rhythmic variations for the chords. They

have been used to generate harmonizations in pop and jazz style or a blend of styles [KP+23].

Methods based on Markov chains are simple and efficient whereas methods based on hidden Markov models are more complex. Both types of methods can generate chord variations due to its probabilistic state transition and output production. Low-order Markov models (e.g., order ≤ 3) can represent local statistical properties, which enable them to capture new short-term musical structure such as the relationship between neighbouring pitches.

In theory, high-order Markov models can capture the long-term structure of chord sequences, but their space and time complexities are too large to be practically viable. For example, high-order Markov chains produce high-dimensional transition matrices with many empty entries and they need a lot of training examples to fill the matrix entries. Moreover, there is no guarantee that the same chord progression can be reproduced in repeated sections. For the same reason, they are not practically viable for capturing long-term or formal structure for harmonization. Markov chains can also work with a single training music piece, but in this case they tend to produce state transition matrices with many empty entries.

3.1.5 Direct Optimization

Direct optimization methods [WC16b; Yi+22; KK23] for chord generation produce chord sequences by directly optimizing an objective function using optimization algorithms. Typically, they represent a note and a chord as a discrete item, although continuous numerical representation is possible. For harmonization, they tend to find the best possible set of chords for a given melody by directly optimizing an objective function that measures the quality of the generated chord-melody matching and chord progression. The objective function can include a variety of factors, such as the harmony, rhythm, and timbre of the chords.

Existing works based on direct optimization include [WC16b] and [Yi+22]. The harmonizer in [Yi+22] optimizes the weighted-sum of note-level chord-melody matching and phrase-level chord progression, whereas the harmonizer in [WC16b] optimizes the weighted-sum of chord-melody matching, chord transition probability and emotional valence value. The harmonizer in [KK23] uses a heavily parameterised

scoring function based on expert knowledge to determine the best chord transitions. These methods apply dynamic programming (DP) to solve for the optimal chord sequence that matches the melody best. Unlike deep neural networks and Markov models, their DP algorithms do not require training.

Direct optimization is a flexible method. The objective function can be designed to reflect specific characteristics of the harmonies. For example, the objective function could favor harmonies that are in a specific key, that have a certain level of complexity, or that are similar to a particular style of music.

While direct optimization is a powerful technique for harmonization, existing methods have some limitations. One limitation is that it is difficult to define a single objective function that generalizes well to various music styles or aesthetic preferences. For example, in [Yi+22], minimizing the dissonance between the melody note and the bass note of a chord is suited to pop music that uses basic major and minor triads. This approach may not be as applicable to jazz, which often features a mix of dissonant extended chord—such as seventh and ninth chords—as well as consonant chords. Secondly, a single objective function cannot cater to other features that are important for harmonization, such as cadence and correspondence to melody structure. Finally, the harmonization output from direct optimization is directly dependent on the chord transition model used. Novel chords with other pitch combinations cannot be generated from direct optimization alone.

3.1.6 Summary

Existing chord generation methods can be compared according to five aspects: training required, amount of example music needed, style specificity, cadence and voice leading, and temporal structure (Table 3.2). Chord-melody matching and chord progression are handled by all these methods so they are not included in the table for comparison. *Training* indicates whether or not a method has to be trained with specific training set. Methods that require training include hidden Markov models and deep neural networks. Markov chains, rule-based methods, genetic algorithms and direct optimization methods do not require training. Markov chains use chord transition probabilities that are pre-computed from a set of example music. Rule-based methods require manual creation of rules and grammars. Genetic algorithms

Table 3.1: Comparison of methods for harmonization.

Methods	Training required	Example music	Style specificity	Cadence, Voice leading	Temporal structure
Rules & Grammars	no	0 or more	difficult	no	short-term, long-term, formal
Genetic Algorithms	no	many	difficult	no	short-term, long-term
Deep Neural Networks	yes	many	tedious	no	short-term, long-term
Markov Chains	no	1 or more	possible	no	short-term
Hidden Markov Models	yes	1 or more	tedious	no	short-term
Direct optimization	no	1 or more	possible	no	short-term, long-term, formal
Proposed method	no	1 or more	easy	yes	short-term, long-term, formal

require at least one pre-defined cost function for evaluating chord selection. Existing direct optimization methods apply dynamic programming to match the chords with the melody.

Example music refers to the amount of example music pieces required for harmonization. They do not necessarily refer to training examples, which imply the need for training. Rule-based and grammar-based methods may not require any example music pieces if the rules and grammars are defined based on music theory. Markov models and direct optimization methods need at least one example music piece as a reference for suitable chord-melody matching. Genetic algorithms and deep neural networks need many examples. In particular, deep neural networks require large datasets of more than 500 music pieces for effective training.

Style specificity indicates whether or not a method can generate a chord sequence that matches a specific music style. Music style refers to different genres of music such as pop, jazz, hard rock and classical. For rule-based and grammar-based methods, it is difficult to manually create rules and grammars for various different

styles. For genetic algorithms, it is difficult, if not impossible, to define cost functions that are specific to a particular style. For Markov chains, it depends on how the chord transition probabilities are computed. It is easy to compute chord transition probabilities based on example music of a specific style. On the other hand, it is difficult to define chord transition probabilities based on neo-Riemannian principles. Thus, it is possible for Markov chain methods to achieve style specificity if the correct approach is adopted. Deep neural networks and hidden Markov models require manual preparation of training examples, which is tedious and time-consuming. To achieve style specificity, they need to train separate versions that each handles a specific style. In principle, direct optimization methods can incorporate reference inputs or constraints in the optimization process to achieve style specificity. Unfortunately, existing direct optimization methods do not adopt any approach that can make them style specific.

Cadence and voice leading denote the ability to guarantee specific chord functions at phrase endings and control over voice leading in a chord sequence. These properties are based on music theory. In principle, all categories of methods, except deep neural networks, rule-based and grammar-based methods, can handle cadence as a hard constraint at phrase ending or section ending. Additionally, they can seek to minimize voice fluctuation in the process of generating chords. Unfortunately, existing methods have not considered cadence and voice leading in harmonization.

Temporal structure describes the ability of a method to produce harmony with regard to short-term, long-term and formal music structure. Most existing methods are able to generate harmony with short-term and long-term structure. Markov models are an exception. To handle long-term structures, they require very large transition matrices that often have too many empty entries, leading to incomplete results. Besides rules, grammars, and direct optimization methods, other methods cannot capture and handle global music structure over the entire music piece. They lack constraints needed to enforce larger structural elements like section boundaries across chord sequences. A notable exception is the method of [Con18] that combines Markov chains with structural rules. The structural rules allow the method to handle global music structure.

My proposed method of harmonization overcomes the shortcomings of existing methods. It applies multi-objective optimization that does not require training.

Therefore, the costs of cleaning, annotating and training on large datasets are avoided. It achieves style specificity by producing chord sequences that match the style of a selected set of example music. In addition, by applying multi-objective optimization, it can handle voice leading as an additional objective and enforce cadence as a hard constraint. It can also handle long-term and formal structure of the given melody according to pre-defined structure labels.

3.2 Accompaniment Generation

Methods for generating accompaniment patterns can be divided into four categories, namely rule-based methods, genetic algorithms, deep neural networks and hybrid methods. Methods involving chordal accompaniment [Niu23] are excluded from this section as it is trivial and does not contain complex textures and rhythm.

There is a related area called automated accompaniment that involves score following [GD97; OD01; Rap04; SOS04; Kle05; PB05; BLB07]. This category of works addresses the playback of accompaniment from an existing score that must be appropriately sped up or slowed down to match the performance of a melody soloist. Although similar techniques are used to match accompaniment to melody, it does not create new accompaniment patterns. Hence, works in automated accompaniment for score following are omitted from this section.

3.2.1 Rule-based and Grammar-based Methods

Methods in this category use rules and grammar to generate accompaniment that consists of single or multiple tracks. These rules and grammar represent the musical features that contribute to the rhythmic, textural or emotional aspects of music accompaniment. They may be manually defined [CLC13; WC16b] or learnt from a training set [Spa99].

The variability and style specificity of the accompaniment generated by these methods is highly dependent on the rules and grammars. Some methods rely on a small set of rules that map music emotion to musical features in existing accompaniment templates to produce emotional accompaniment [CLC13; WC16b]. Other works used extensive sets of rules and grammars based on one target style such as Bach [Spa99] and pop [SDS10] to achieve multi-track accompaniment.

The advantage of rule-based and grammar-based methods is that accompaniment targeting pop and classical music can be achieved using rules and grammars guided by traditional music theory. However, these methods can neither generalize well to other styles nor allow for personalized styles outside of music theory without any rule violation [WC16a]. It is difficult to create a comprehensive set of rules that cover all possible accompaniment styles. While this shortcoming can be overcome using a probabilistic combination of general and style-specific rules, this approach requires those probabilities to be learnt [Spa99] or the justification of using manually selected probabilities [SDS10].

3.2.2 Genetic Algorithms

Genetic Algorithms (GAs) for accompaniment generation [Dos05; LT12] rely on one or more fitness functions to generate and evaluate newly generated accompaniment variations to suit an existing melody. The fitness function describes the goodness of fit for accompaniment-melody matching. Similarly to GAs for harmonization, they repeat a cycle of evaluation, selection and reproduction with variation. In the evaluation stage, candidate solutions generated from user-specified examples or a random process are evaluated using a fitness function that captures accompaniment-melody phrase matching. In the selection stage, a new set of candidate accompaniment phrases are probabilistically copied from the old set, in proportional to its fitness. In the reproduction stage, some operators are applied to the set of candidates to generate new candidate accompaniment phrases. This three-stage evolution cycle repeats until the accompaniment with high fitness is obtained.

GA methods for accompaniment generation differ mainly in the choice of music theory rules that inform the fitness function. In [Dos05], the fitness function evaluates the match between drum grooves and a non-melodic phrase to produce a drum accompaniment. In [LT12], three fitness functions are designed to generate the chordal accompaniment, bass part and final chord arrangement. The first fitness function determines chord tone, non-chord tone and rests for the given melody as the main chordal accompaniment. The second fitness function utilizes the chord information from the chordal accompaniment to generate a matching bass line. The

last fitness function determines the basic chord arrangement to be used among four templates, given the chordal accompaniment and bass part.

With existing GAs, accompaniment variations with short-term and long-term structure can be obtained by exploring various combinations of accompaniment phrases that match the given melody. However, there are some shortcomings. Firstly, there is no guarantee that the accompaniment progression is smooth across phrases. Secondly, it is difficult to encourage correspondence between the accompaniment and the formal structure of melody as these fitness functions do not consider music structure. Finally, it is difficult to define an appropriate fitness function that covers a set of music rules that suit various accompaniment styles.

3.2.3 Deep Neural Networks

Deep Neural Networks (DNNs) for generating accompaniment [Hua+17; LY18; LWY18; Zhu+18; Zhu+20; Wan+20; Hsi+21; Yi+22] have gained traction in recent years. Similar to their application in chord generation, the DNNs for accompaniment generation encode notes and chords as numeric values, and generate polyphonic accompaniment output. During training, they learn a highly complex non-linear mapping between melody and one or more accompaniment tracks, using stochastic gradient descent and backpropagation. During the generation phase, features extracted from the input and the features learned from training examples are processed via feedforward neuron activation to generate polyphonic output.

Families of DNNs used for accompaniment generation include Convolutional Neural Networks (CNNs), Generative Adversarial Network (GANs) and Transformers. The CNNs for accompaniment generation applies cross-correlation on polyphonic training examples to generate accompaniments in Bach style [Hua+17]. GANs simultaneously train two neural networks with existing music and random noise to synthesize new music that closely resemble training examples. In [LY18; LWY18], GANs are used to generate polyphonic accompaniment that matches a given lead sheet melody, whereas the GAN in [Zhu+20] generates multi-track accompaniment for melodies targeting pop, classical and jazz styles but require re-training to switch between styles. Transformer networks incorporate a self-attention mechanism that relate different positions of a single sequence in its encoder and decoder. In [Hsi+21],

Transformers are used to generate accompaniment from lead sheet and corresponding polyphonic piano arrangements in an autoregressive manner. [Zhu+18] also use a similar attention mechanism but with significantly fewer neurons to generate multi-instrument accompaniment in pop style.

The major strength of DNNs is that the compositional principles of accompaniment such as texture and voice leading need not be explicitly defined to generate polyphonic accompaniment. This is because the relationships between melody and single or multi-track arrangements are captured implicitly as learnt weights during training. With a large training set representing the target accompaniment that belong to a particular genre or similar accompaniment styles, trained DNNs can generate accompaniments variations in the desired style.

DNNs for accompaniment generation have several shortcomings. Firstly, DNNs require large training sets, which can be expensive to collect [JYL24]. Secondly, they cannot ensure style specificity unless positive examples outweigh negative examples [Zhu+20]. Finally, DNNs cannot directly generate structured accompaniment that corresponds to the input melody structure. They require other methods to impose structural constraints [Wan+20; Yi+22].

3.2.4 Markov Models

Only one existing work uses Markov model to generate accompaniment [Mo22]. It consists of two hidden Markov models (HMMs). The first HMM generates the rhythm of accompaniment given an input melody. The second HMM generates a chord sequence given the generated accompaniment rhythm. Finally, the Viterbi algorithm is applied to produce a sequence of accompaniment phrases, based on a set of example accompaniment phrases, that match both the generated rhythm and generated chord sequence

The main advantage of HMMs is that it can generate appropriate output sequences after being trained on large set of training examples. However, HMMs are unable to generate accompaniment with long-term or formal structure, unless the training set contains long-term phrases. In addition, high-order HMMs require large transition matrices that can contain many empty entries. Consequently, there is no guarantee that a complete accompaniment solution can be found.

3.2.5 Hybrid Methods

This category of methods combines deep neural networks (DNN) with other methods to generate accompaniment for melody. The method in [LZ22] consists of a DNN and a HMM. The DNN takes the melody as input and generates a sequence of chords labels. The HMM takes the generated chord labels as inputs and generate accompaniment. In contrast, the method in [ZX21] uses a HMM followed by a DNN. The HMM uses the Viterbi algorithm, a variant of dynamic programming, to produce a globally optimal sequence of chords for the given melody. The DNN is trained on pairs of reference accompaniment and chords. During the generation phrase, the DNN takes a reference accompaniment, reference chords and chords generated by the HMM as input and rearranges the accompaniment to suit the generated chords.

Hybrid methods that combine DNN with other methods can outperform single methods on a complex problem such as accompaniment generation. They manage the complexity by using existing single methods to separately handle sub-problems such as chord-melody matching and accompaniment generation. In particular, the use of Dynamic Programming in [ZX21] can produce globally optimal solution for chord generation.

Nevertheless, hybrid methods still have shortcomings. Due to the use of DNN, the tedious and expensive process of collecting large training sets, and the time-consuming training procedure, cannot be avoided. HMMs also require training although it can be done with smaller training sets.

3.2.6 Summary

Accompaniment generation methods can be characterized according to seven aspects, namely, training required, amount of example music needed, style specificity, novelty, voice density, voice interaction and temporal structure (Table 3.2). Chord-melody matching and rhythm matching are handled by all these methods, so they are not included in the table for comparison.

Training indicates whether or not a method has to be trained with a specific training set. Methods that require training include hidden Markov models and deep neural networks. In contrast, rule-based and grammar-based methods do not require

training, but require manual creation of rules and grammar.

Example music refers to the amount of example music pieces required for accompaniment generation. They do not necessarily refer to training examples, which imply the need for training. Rule-based and grammar-based methods may not need any example music piece if they are defined based on music theory. Markov models need at least one example music that encapsulates suitable accompaniment-melody matches. Genetic algorithms and DNNs need many examples for accompaniment generation. In particular, DNNs require large datasets of more than 500 music pieces for effective training.

Style specificity refers to the ability of a method to generate accompaniment that matches a specific music style. For rule-based and grammar-based methods, it is difficult to manually create rules and grammars for various accompaniment styles. For genetic algorithms, it is difficult, if not impossible, to define cost functions that are specific to a particular accompaniment style. Deep neural networks and hidden Markov models require manual preparation of training examples, which is tedious and time-consuming. To achieve style specificity in accompaniments, they need to train separate versions that each handles a specific style.

Novelty indicates the ability to produce accompaniment phrases that are sufficiently different from those in the example music. The generated accompaniment is considered novel if its texture and voice leading patterns are different from those in the example phrases. Note that just changing the chords of accompaniment phrases taken from example music does not constitute novelty because their texture and voice leading patterns are unchanged. There is no known method that can directly control accompaniment novelty.

Voice density refers to the average number of notes in a specific time span, which influences the sense of accompaniment flow. *Voice interaction* refers to the degree of voice leading and harmonic dependence between multiple voices. It is difficult to define rules and grammars that describe voice density and voice interaction. Genetic algorithms, Markov models and deep neural networks cannot control voice density and voice interaction in accompaniment. An exception is the hybrid HMM+DNN method of [ZX21]. It controls voice density by pre-filtering reference accompaniments that do not match the desired density, based on the average number of notes played simultaneously.

Temporal structure refers to the ability of a method to produce accompaniment with regard to short-term, long-term and formal music structure. Most existing methods can generate accompaniment with short-term and long-term structure. Markov models are an exception. To handle long-term structure, Markov models require very large transition matrices that may contain many empty entries, leading to an incomplete results. Besides rules and grammars, other methods cannot handle formal music structure.

My proposed method of accompaniment generation overcomes the shortcomings of existing methods. It applies multi-objective optimization that does not require training. It achieves style specificity by producing accompaniment that matches the style of a selected set of example music. Therefore, the costs of cleaning, annotating and training on large datasets are avoided. In addition, by applying multi-objective optimization, it should be able to handle voice leading, voice interaction and voice density as additional objectives. It can also handle long-term and formal structure of the given melody according to pre-defined structure labels.

Table 3.2: Comparison of methods for accompaniment generation.

Methods	Training required	Example music	Style specificity	Novelty	Voice density	Voice interaction	Temporal structure
Rules & Grammars	no	0 or more	difficult	no	no	no	short-term, long-term, formal
Genetic Algorithms	no	many	difficult	no	no	yes	short-term, long-term
Deep Neural Networks (DNN)	yes	many	tedious	no	no	no	short-term, long-term
Hidden Markov Models (HMM)	yes	1 or more	tedious	no	no	no	short-term
DNN + HMM [LZ22]	yes	many	tedious	no	no	no	short-term
HMM + DNN [ZX21]	yes	many	tedious	no	yes	no	short-term
Proposed method	no	1 or more	easy	yes	yes	yes	short-term, long-term formal

3.3 Performance of Deep Neural Networks for Music Generation

Two studies have been conducted to compare the performance of deep neural networks and Markov models in generating monophonic and polyphonic music. They investigate the quality of generated 30-second melodic phrases [Yin+21; Yin+23] and polyphonic music in the style of a classical string quartet [Yin+23].

The study by [Yin+23] indicates that existing deep neural networks show negligible or no improvement in performance compared to Markov models in terms of stylistic conformance to a target music style. For instance, according to [CL17], the stylistic conformance ratings achieved by the MAIA Markov model is comparable to those achieved by the state-of-the-art DNNs such as MusicVAE [Rob+18] and Music Transformer [Hua+19]. Consistent with the results of [Hua+19], the study reported significantly higher stylistic conformance ratings for human-composed music compared to auto-generated music.

The authors of [Yin+21] identified a noticeable lack of originality in the music generated by deep neural networks. They define originality as the degree to which an algorithm copies from example music pieces on which it is trained. To assess originality, the authors conducted tests on melodies generated by Music Transformer [Hua+19] and MAIA Markov model [Rob+18]. They also established a baseline as the degree of copying in human-composed melodies with respect to an existing set of melodies. Test results revealed that 2-bar and 4-bar melodies generated by Music Transformer exhibited a lower degree of originality compared to the baseline. In contrast, the melodies produced by MAIA Markov model demonstrated similar levels of originality to those of human composers. Further analysis indicated that the Music Transformer tended to reproduce exact copies of training phrases, particularly when the network was trained for a long period of time.

Both studies conclude that the performance of existing deep neural networks in terms of stylistic conformance and originality are no better than that of a well-trained Markov model. They advocate caution to the prevalent view of deep learning as the primary approach to automated music generation.

Chapter 4

Harmonization

The goal of this research consists of two parts, namely harmonization and accompaniment generation. This chapter presents a method for harmonization called FlexChord that is completed as the preliminary work of the thesis proposal. In automated music composition, music needs to be represented in a form that facilitates computation. Thus, this chapter begins with the computational representation of music elements.

4.1 Representation of Music Elements

A melody consists of a sequence of notes and rests. A note has a pitch value that is encoded as a MIDI note number ranging from 0 to 127, which corresponds to C1 and G9 respectively. A rest has no pitch value. Nevertheless, for computational convenience, a rest is often represented by the pitch value of 0. This is acceptable because pitch value 0 has very low frequency and is rarely used in music. The melody also includes other attributes, namely key K in major or minor mode, time signature S , and the number of pitches per beat β , which is 4, 8, 16 or 32.

The relationship between two pitches can be described and measured in several ways. The signed difference between two pitches p and q is called **pitch difference** d_p :

$$d_p(p, q) = q - p. \quad (4.1)$$

With p and q in MIDI note numbers, this difference measures the number of semitones between p and q . The **pitch interval** $\theta(p, q)$ between pitches p and q is defined as

$$\theta(p, q) = |p - q| \bmod 12.$$

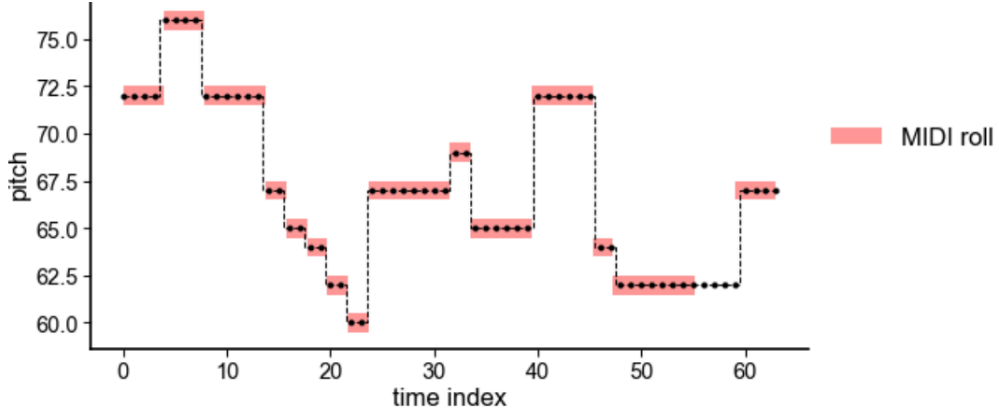


Figure 4.1: Step function of a melody. Sustained pitches are highlighted in red. Horizontal lines that are not highlighted indicate rests.

Two pitches that are separated by exactly one octave share the same pitch class, which is an integer between 1 and 12. Thus, **pitch class** pc of pitch p is computed as

$$pc(p) = p \bmod 12. \quad (4.2)$$

In computation, melody is typically represented as a step function (Fig. 4.1) of m pitches over integer time index $i = 0, \dots, m - 1$. Each pitch p_i at time index i is encoded by a MIDI pitch number. A change of pitch corresponds to a pitch onset. Rests are represented separately outside the step function. The horizontal part of a step function between a pitch onset and an immediately following onset or rest corresponds to the duration of the pitch.

The objective of harmonization is to generate a chord at each **strong beat** to match the pitch sequence and structure of a given melody. Let j denote strong beat index, for $j = 0, \dots, n - 1$. It is related to time index i by

$$j = \text{int}(i/\alpha\beta), \quad \alpha = \begin{cases} 2 & \text{if } S = 2/4, 4/4, \\ 3 & \text{if } S = 3/4, 6/8, \end{cases} \quad (4.3)$$

where β is the number of pitches per beat, which is 4, 8, 16 or 32, and S is the key signature of the melody. This gives 1 strong beat per bar for 2/4 and 3/4 music and 2 strong beats per bar for 4/4 and 6/8 music. With strong beat index j , the

Table 4.1: Pitch dissonance depends on pitch interval [Hur94].

Pitch Interval $\theta(p, q)$	Pitch Dissonance $d(p, q)$	Example: $p = C$
0	0	C
1, 11	1	C#, B
2, 10	0.8	D, A#
3, 9	0.5	D#, A
4, 8	0.4	E, G#
5, 7	0.3	F, G
6	0.7	F#

melody is organized as a sequence of fragments h_j , $j = 0, \dots, n - 1$. Each fragment h_j contains pitches $p_{\alpha\beta j}, \dots, p_{\alpha\beta(j+1)-1}$. It has a structure label λ_j that takes the value of ‘P’, ‘S’, or ‘ ’ respectively for end of phrase, end of section which is also end of phrase or neither. A phrase is a 4-bar or 8-bar sequence that is no longer than a section. For a fragment h_j that is not the end of phrase and section, and repeats an earlier phrase $h_{j'}$ exactly, its $\lambda_j = j'$.

The objective of harmonization is to produce chords c_i , each consisting of multiple pitches q_{ik} , for $k = 1, \dots, \kappa$, where κ is the number of pitches or voices in the chord. Chord c_j at strong beat j is matched to fragment h_j of the melody, and it is held (i.e., kept unchanged) for the duration from $i = \alpha\beta j$ to $\alpha\beta(j + 1) - 1$. Thus, it is required that $m = \alpha\beta n$. Only chords c_j have onsets; the others are sustained.

4.2 Problem Formulation

Harmonization is the process of forming chords to support and enhance a given melody. It needs to fulfill various aspects namely chord-melody matching, chord progression, cadence, and voice leading. These aspects are formulated as computational objectives and constraints.

4.2.1 Chord-Melody Matching

Music perception studies point to the match between a chord c_j and a melody fragment h_j as the amount of dissonance between their pitches [Hur94; TDSG18]. According to Huron’s study [Hur94], the dissonance value $d(p, q)$ between two pitches

p and q depends on their pitch interval $\theta(p, q) = |p - q| \bmod 12$ (Table 4.1). The higher the dissonance, the less harmonious are p and q when they are played together.

A similar concept can be defined for the dissonance $d(h_j, c_j)$ between fragment h_j and chord c_j . It is measured by the average dissonance between their pitches:

$$d(h_j, c_j) = \frac{1}{|h_j||c_j|} \sum_{p_i \in h_j} \sum_{q_{jk} \in c_j} d_p(p_i, q_{jk}), \text{ for } p_i \neq \text{rest}, \quad (4.4)$$

where $|h_j|$ and $|c_j|$ are the total number of pitches in melody fragment h_j and chord c_j respectively. [Yi+22] uses the same concept but with slightly different dissonance values.

4.2.2 Chord Progression and Music Style

In harmonization, music style is conveyed by chord progression. Let c and c' denote consecutive chord pairs in a reference music, where c' immediately follows c . Then, chord transition probability $P(c' | c)$ can be computed by measuring the frequency of occurrence of chord pair (c, c') in the reference music. These transition probabilities form the chord transition model C that captures the style of the reference music.

The chords in chord transition model C are represented by Roman numerals (Section 2.1). This representation is independent of music key and actual chord positions. Thus, the chord transition model C can be applied to the harmonization of melodies of any key. In contrast, most neural network methods represent each position of the same chord (e.g., C-E-G, E-G-C) as a different chord. [Yi+22] also uses Roman numerals to represent chords, but it limits its chord types to major and minor triads. In contrast, this research work uses all chord types.

4.2.3 Cadence

In music theory, cadence refers to melody or harmony at the end of a phrase or section that creates a sense of full or partial resolution [Ran99]. Different cadences create different senses of resolution. They help to clarify the overall hierarchical structure of music [Tem11].

In practice, it remains difficult to define a fixed set of cadences across multiple music styles [Cap04]. A simplified approach is to refer to cadence as specific chord-pairs at the end of a phrase or a section. Each chord pair (c, c') in chord transition

model C is given a structure label that takes the value ‘P’, ‘S’ or ‘ ’, as defined for melody fragments h_j (Section 4.1). Then, cadence can be handled by matching melody fragments labeled ‘P’ and ‘S’ to chord pairs with the same labels (see Section 4.2.5 for details).

4.2.4 Voice Leading

Voice leading refers to the interaction of individual voices that creates harmony [Hur16]. It is an important aspect of music where multiple voices play supporting roles in held chord generation. According to the pitch proximity principle, a chord progression that minimizes voice fluctuation has a good or *efficient* voice leading, which improves the coherence of the music [Hur01; Hur16]. The voice fluctuation between two successive chords can be defined as the average distance moved by the implied voice parts between them. For two chords $c_i = \{q_{ik}\}$ and $c_{i-1} = \{q_{i-1,k}\}$, the voice fluctuation δ_i between them is defined as

$$\delta_i = \begin{cases} \sum_{k=1}^{\kappa} |q_{ik} - q_{i-1,k}| & \text{for } q_{ik} \neq \text{rest and } q_{i-1,k} \neq \text{rest} \\ 0 & \text{otherwise.} \end{cases} \quad (4.5)$$

4.2.5 Constraint and Optimization Objectives

The conditions of harmonization discussed in the previous sections are translated into a hard constraint for cadence and three optimization objectives:

1. Chord progression and style matching:

Style is captured in the chord transition model C . By maximizing the chord transition probability $P(c_j | c_{j-1})$ of the selected chords c_j , for $j = 1, \dots, n-1$, the generated chords c_j follows the style of C . Note that chord c_0 has no previous chord.

2. Chord-melody matching:

By minimizing overall chord-melody dissonance D :

$$D = \sum_{j=0}^{n-1} d(h_j, c_j), \quad (4.6)$$

the generated chords will match the melody well in terms of harmonization.

3. Smooth voice leading:

Minimizing overall voice fluctuation Δ produces smooth chord voices:

$$\Delta = \sum_{i=1}^{m-1} \delta_i. \quad (4.7)$$

For held chords, voice fluctuation δ_j can also be defined between the chords at successive strong beat indices $j - 1$ and j because the chords between them remain unchanged.

4.3 Multi-Objective Optimization

4.3.1 Overview

In general, a problem with multiple objectives cannot be perfectly solved by optimizing all the objectives simultaneously. A principled way of handling more than one objective is to apply multi-objective optimization [Ehr05; Mie99]. This family of methods generalizes the optimality condition of single-objective optimization to the condition of **Pareto optimality**. A Pareto optimal solution is a solution such that no objective function can be further improved without compromising other objective functions. A multi-objective optimization problem can have many Pareto optimal solutions. The set of all Pareto optimal solutions defines the **Pareto front**. A good method should be able to produce all possible Pareto optimal solutions, for example, by varying internal parameters randomly.

One commonly used approach for multi-objective optimization, called linear scalarization, combines multiple objective functions into a single function by a weighted sum. Then, the single weighted function is optimized using standard methods. This approach is used by many solution methods including Markov models and deep neural networks. Despite its simplicity, it has an inherent shortcoming—it cannot produce some Pareto optimal solutions when the Pareto front is non-convex [Ehr05; Mie99]. In other words, these methods can miss many good solutions regardless of how well they are designed and trained.

An alternative to linear scalarization is the lexicographic method [Ehr05; Mie99]. It first minimizes the most important objective function. If this step produces a set of

optimal solutions, then it looks for solutions in the set that minimizes the next most important objective function. This procedure is continued until a unique solution remains or after all objective functions have been optimized. A shortcoming of the lexicographic method is that less important objective functions are minimized only when there are multiple optimal solutions or ties. If there is no tie, less important objective functions are never minimized—they are effectively ignored. In addition, lexicographic methods can only produce a very small subset of solutions on the Pareto front.

The lexicographic method can be slightly modified to overcome its shortcomings [Mie99]. In each step, instead of looking for solutions that minimize an objective function, it looks for solutions that are good enough, i.e., solutions that are close to and including the minimum of the objective function. The bounds that specify closeness are manually set. When the bounds are small, the Pareto solutions obtained are very near to the minima of the first few objective functions. On the other hand, when the bounds are large, the Pareto solutions are far from the minima of the first few objective functions and nearer to those of the last few objective functions. Therefore, by varying the bounds, the modified lexicographic method can produce all solutions on the Pareto front [Leo24].

In the harmonization problem formulated in Section 4.2, time is discretized into integer index i . Strong beat index j is an integer and a chord is a discrete item. In other words, the solution space of harmonization is discrete. Therefore, the algorithm for solving the harmonization problem has to be a discrete optimization algorithm. Standard constrained optimization methods for the continuous domain are not applicable.

4.3.2 FlexChord Harmonization

FlexChord (Algorithm 1) adopts a discrete version of the modified lexicographic method (Lines 3–4, 14–15). Its overall structure is to assign a chord to the first beat of each fragment and hold the chord for the duration of the fragment length of $\alpha\beta$ by repeating the same chord without onset (Line 17). The first chord of the melody, with $j = i = 0$, is set to a chord c from the chord transition model C having a high transition probability and low dissonance with the first fragment h_0 (Lines 2–5).

For a fragment h_j at $j > 0$, several cases are considered. If h_j is just before a phrase or section ending labeled ‘P’ or ‘S’, a chord-pair (c, c') that matches structure label λ_{j+1} and has the lowest combined dissonance is chosen. Chord c is assigned to c_j , and c' is assigned to c_{j+1} (Lines 6–10). These steps satisfy the cadence constraint. Chord c_j is set at a position with the least voice fluctuation and c_{j+1} is set at the root position as cadence. The actual chord is determined by the chord’s Roman numeral and the melody’s key. For example, chord I is C major chord for C major key, G major chord for G major key, etc.

If λ_j is an integer j' , it means that fragment h_j repeats $h_{j'}$ exactly. So, chord $c_{j'}$ is copied to chord c_j (Line 11–12). This method generates the same chord sequence for repeated phrases and takes care of the long-term structure of repeating phrases. An alternative to this approach is to begin each phrase in the same manner as the beginning of melody (Lines 2–5). Then, being deterministic, FlexChord will generate the same chord sequence for the repeating phrase. This approach sacrifices the voice leading at the beginning of the earlier repeated phrase.

Otherwise, K chords having the largest transition probabilities are selected (Line 14). This step maximizes chord transition probability (Objective 1). Among these K chords, the chord c with the least dissonance is selected (Line 15). This step minimizes chord-melody dissonance (Objective 2). When $K = 1$, the algorithm ignores Objective 2. When $K = N$, it ignores Objective 1. A good balance between Objectives 1 and 2 is found with $1 < K < N$. This balancing of Objectives 1 and 2 are also applied to the first chord at $j = 0$ (Lines 3–4). A chord selected in Line 15 can have several chord positions (Section 2.1). Line 16 sets chord c_j at a position of c that minimizes voice fluctuation (Objective 3). Objective 3 is omitted at end of phrase and section to cater to cadence. Therefore, FlexChord performs modified lexicographic optimization of 3 objective functions and 1 constraint.

Note that the chord transition model C can be prepared from a large set of reference music pieces (Section 4.4.1) or a single reference music piece (Section 4.4.5). A small reference set may contain chords without pairing to other chords. Such chord-pairs are assigned 0 transition probability in the chord transition model. Thus, the chord transition model contains all possible pairings of chords for each chord type present in the reference set. Chord-pairs with 0 transition probability may be selected if it is among the top K pairs with the highest transition probability. This

ensures that the selection of top K chords in Line 3 and 14 is always successful.

Algorithm 1: FlexChord Harmonization

Input: Melody M , style-specific chord transition model C .
Output: Chords c_i , $i = 0, \dots, m - 1$.

- 1: **for** $j = 0, 1, \dots, n - 1$ **do**
- 2: **if** $j = 0$ **then**
- 3: Select K chords with the largest transition probabilities to any chord.
- 4: Among K chords, select chord c with the least dissonance $d(h_0, c)$.
- 5: Set c_0 at the root position of c .
- 6: **else if** $\lambda_{j+1} = 'P' \text{ or } 'S'$ **then**
- 7: Select a matching chord-pair (c, c') with the lowest combined dissonance.
- 8: Set c_j at the position of c with the least voice fluctuation δ_j .
- 9: **else if** $\lambda_j = 'P' \text{ or } 'S'$ **then**
- 10: Set c_j at the root position of c' .
- 11: **else if** λ_j is an integer j' **then**
- 12: Set $c_j = c_{j'}$.
- 13: **else**
- 14: Select K chords with the largest transition probabilities $P(c | c_{j-1})$.
- 15: Among K chords, select chord c with the least dissonance $d(h_j, c)$.
- 16: Set c_j at the position of c with the least voice fluctuation δ_j .
- 17: Set $c_i = c_{i-1}$, $i = \alpha\beta j + 1, \dots, \alpha\beta(j + 1) - 1$.
- 18: Set chord onset for chord c_j .

4.3.3 Analysis of Time Complexity

FlexChord runs for exactly n iterations, one for each strong beat index j , to produce n chords. In each iteration, lines 3, 4, 14 and 15 perform discrete search for appropriate chords. Discrete search can be implemented in several ways with different time complexity. Linear search has a time complexity of $O(N)$, where N is the number of chords in the chord transition model C . Indexed lists of chords sorted in decreasing order of transition probability has time complexity of $O(1)$. Lines 3 and 14 select K chords among all chord types in the chord transition model C , and Lines 4 and 15 select one chord among the K chords. Since K is a small constant parameter, it has no significant effect on time complexity. Line 7 selects a chord pair with appropriate structure label. The number of chord pairs with structure labels is small compared to all possible pairings of chords. It is estimated to be of

order $O(N)$. Therefore, the time complexity of FlexChord is either $O(nN)$ or $O(n)$ depending on the implementation of discrete search.

The number of iterations n depends on the length of the melody. In the experiments (Section 4.4), the value of n ranges from 16 to 124. The number of chords N used in the test is less than 100 (Table 4.2). Therefore, the actual execution time of FlexChord is very short.

4.4 Experiments and Discussions

This section presents two types of tests: short-phrase test and full-length test. Short-phrase test evaluates FlexChord’s performance on 8-bar phrases for comparison with deep neural networks. Both quantitative and qualitative test results are discussed. Full-length test evaluates FlexChord’s performance on complete songs. Qualitative test results are discussed to reveal details of chord generation.

4.4.1 Data Preparation

Two public datasets, namely Chord Melody Dataset (CMD) [Hie19] and Hook Lead Sheet Dataset (HLSD) [And21] were adapted for the harmonization task. The original datasets contain a mixture of pop, jazz, classical and new age music. Algorithms that are trained on these datasets could produce arbitrary results that reflect a mixture of genres as opposed to a specific target genre. To avoid such confusion, two large sets of songs of distinct genres, namely pop and jazz, were extracted from the combined pool of CMD and HLSD resulting in 1000 pop songs and 266 jazz songs respectively. This selection of songs catered to a substantial number of training examples for a fair comparison with recent deep neural networks. A smaller set of 30 pop songs randomly selected from the pool was also prepared in order to assess if there were any performance differences between the use of small and large pop datasets.

Jazz songs were selected to contrast with pop songs due to their greater chord and harmonic complexity. They contained extended chords and frequent key changes. In contrast, pop harmonies consisted of triads or basic seventh chords and were less adventurous with chord progressions.

The deep neural networks that were used for comparison worked only on songs

Table 4.2: Summary statistics of the datasets used for a comparative study.

Dataset	Number of Songs	Number of 8-bar phrases	Number of unique chord types, N
Pop-S	30	2867	43
Pop-L	1000	81,625	70
Jazz-L	266	17,969	71

with 4/4 time signatures. Thus, only such songs were selected for testing. Similar to existing work, their tempo was set to 120 beats per minute and they were transposed to all 12 music keys to ensure that the neural networks could handle test melodies of various keys. Each song was encoded in step-function representation with number of pitches per beat β set to 16. A chord was sampled every half-bar over 8 bars of music.

Songs for each dataset were split into 8-bar phrases. 8-bar phrases were used because existing neural networks were trained and evaluated on 8-bar phrases. With 8-bar phrases, deep neural networks could be trained with shorter training time and facilitate easier quantitative evaluation.

Each dataset was split into training, validation (i.e., fine tuning) and testing set at the ratio of 8:1:1. Subsets for neural network training and validation consisted of 8-bar phrases with an overlap of 2 bars. On the other hand, subsets for testing consisted of 8-bar phrases aligned to phrase boundaries without overlap. For testing, each 8-bar phrase in the test set was assigned a structure label where $\lambda_j = \text{'P'}$ for $j = 7, 15$ and $\lambda_j = \text{' '}$ otherwise. Table 4.2 summarizes the statistics of the prepared datasets.

FlexChord does not require training. The training set was used to build the chord transition model of FlexChord and the testing set was used for performance testing. The validation set was not used by FlexChord. For the chord transition model, all chords were represented by Roman numerals. The Roman numerals were captured at the root position, ignoring the actual inversions. Chord transition probabilities were computed from the frequency of occurrence of consecutive chord-pairs in the training set. Structure labels were assigned to the chord-pairs as described in the previous paragraph.

4.4.2 Quantitative Evaluation

(a) Test Procedure

The objective of this test is to investigate the performance of FlexChord (FC). For comparison, a handicapped version of FlexChord called FC– was also tested. In essence, FC– is FC without Lines 6–12. That is, FC– does not handle cadence and is unaware of structure labels. In addition, FlexChord was compared with two well-known deep neural networks, namely **VTHarm** by Rhyu *et al.* [Rhy+22] and **DAT-VAE** by Zhao *et al.* [ZXW22]. Both neural networks have different architectures but use similar loss functions for harmonization. They were trained on the same datasets (Section 4.4.1) with published training parameters [Rhy+22; ZXW22].

VTHarm is a Transformer-based neural network that minimizes chord reproduction error. It also minimizes the Kullback–Leibler (KL) divergence between the learned distribution and target Gaussian distribution of chords. Like the standard Transformer, it has a self-attention mechanism that functions as a weighting scheme for capturing chord similarity at different time points within the sequence. It also has an additional encoder that encodes melody and chords, along with the melody’s key signature.

DAT-VAE encodes the melody and chords with a variational autoencoder network (VAE). The VAE aims to minimize a loss function that consists of 2 terms, namely chord reproduction loss and KL divergence between the learned distribution and target Gaussian distribution of chords. Additionally, it has a preprocessor neural network (DAT) that is trained to map all 12 transpositions of melody to the melody of a canonical key. DAT’s goal is to normalize the key of the melodies so that the whole DAT-VAE can learn to generate suitable chord progressions regardless of the actual key of the melody.

The training objective of VTHarm and DAT-VAE focuses on reproducing training chords and modeling the Gaussian distribution of chords. They do not explicitly maximize chord transition probability and minimize dissonance between the generated chords and the melody. In addition, they use different training inputs and architectures to handle melodies in different keys. VTHarm is specifically trained with melody-chord pairs along with the melody key signatures. It also integrates a

self-attention mechanism that considers chord similarity with other chords in the sequence. In contrast, DAT-VAE does not require melody key signatures at training. Instead, its DAT network normalizes the input melodies to a canonical key.

Both VTHarm and DAT-VAE were trained on the training set, fine tuned using the validation set, and then tested on the testing set. FC and FC– used the chord transition model derived from the training set (Section 4.4.1). They were tested on the testing set without requiring training and fine tuning. The number of candidate chords K was set to 5 for both variants. This procedure was repeated for each of the three datasets of Pop-S, Pop-L and Jazz-L.

(b) Quantitative Test Metrics

For objective comparison between FlexChord and neural networks, a set of simple but musically meaningful metrics were identified:

- Voice Fluctuation V

V is the overall voice fluctuation in the chord sequence (Eq. 4.5). A low V value means that voices in successive chords have lower pitch difference. This enhances the smoothness of chord progression.

- Fraction of inverted chords IC

IC is the fraction of inverted chords in the chord sequence. A high IC value means that a large variety of chord positions beside root position exist within a chord sequence. The presence of chord inversions create movement and they encourage lower voice fluctuation V .

- Chord histogram entropy H

H is the amount of entropy (or disorder) in a chord histogram. It is defined as:

$$H = \sum_k^N p_k \log p_k, \quad (4.8)$$

where p_k denotes the probability of the k^{th} bin of the histogram and N is the total number of chord types present in the sequence. It ignores inversions and the order of chords produced. A higher H indicates a wider variety of chords used.

- Dissonance D

The average dissonance D between melody notes and chords is defined as:

$$D = \frac{1}{n} \sum_j d(h_j, c_j), \quad (4.9)$$

where $d(h_j, c_j)$ denotes the average dissonance between pitches of melodic fragment h_j and chord c_j (Eq. 4.4). Using dissonant chords around a specific melody note can draw attention to it, adding richness to the overall harmony. The lower the D the more harmonious is the match between the chord and the melody. Nevertheless, the appropriate level of dissonance depends on the target music style.

- Pure chord tone to non-chord tone ratio R0

This metric counts the ratio of chord tones n_c to non-chord tones n_n in the melody fragment h_j in relation to its assigned chord c_j :

$$R0 = \frac{n_c}{n_c + n_n}. \quad (4.10)$$

This metric measures the harmony between chord and melody. The higher the $R0$, the better is the match between the chord and melody.

- Chord tone to non-chord tone ratio R1

$R1$ is similar to $R0$, except that proper non-chord tones are counted as chord tones [Yeh+21]. A proper non-chord tone is a non-chord tone p_i that is within two semitones of its next tone p_{i+1} .

$$R1 = \frac{n_c + n_p}{n_c + n_n}, \quad (4.11)$$

where n_p is the number of proper non-chord tones. The higher the $R1$, the better is the match between the chord and melody.

Among these metrics, V , IC and H measure chord diversity, while D , $R0$ and $R1$ measure chord-melody match. In particular, H and $R1$ are part of the canonical metrics [Yeh+21] used in existing DNN-based harmonization works [Rhy+22]. The remaining metrics in [Yeh+21] are omitted because they are related to H and $R1$. Metric $R1$ uses the concept of proper non-chord tone that is not defined in music theory. Therefore, the alternative $R0$ is included along with $R1$. The additional metrics V , IC and D are included to contrast the performance of FlexChord and DNNs.

Table 4.3: Quantitative evaluation for chord diversity and chord-melody match.

		Chord diversity			Chord-melody match		
Dataset	Method	V↓	IC↑	H↑	D↓	R0↑	R1↑
Pop-S	VTHarm	NA	0.00	1.15	6.81	0.80	1.23
	DAT-VAE	NA	NA	1.46	4.25	0.63	0.63
	FC–	0.66	0.61	1.57	5.02	0.86	1.12
	FC	0.90	0.48	1.74	5.01	0.88	1.15
Pop-L	VTHarm	NA	0.00	1.72	6.44	0.83	1.24
	DAT-VAE	NA	NA	1.44	4.36	0.57	0.57
	FC–	0.69	0.52	1.44	5.10	0.89	1.32
	FC	0.97	0.44	1.68	5.21	0.88	1.31
Jazz-L	VTHarm	NA	0.00	2.09	7.27	0.81	1.23
	DAT-VAE	NA	NA	2.00	4.55	0.55	0.55
	FC–	0.63	0.63	1.57	5.75	0.88	1.42
	FC	0.96	0.53	1.91	5.75	0.88	1.41

(c) Quantitative Comparison of Various Methods

Table 4.3 tabulates the quantitative results of all the methods. FC– achieves the lowest voice fluctuation V indicating that it produces chords with the smoothest voices. This is because it directly minimizes voice fluctuation (Lines 8, 16). On the other hand, FC has a slightly higher V than does FC– because it enforces the cadence constraint that fixes chords in root position at phrase or section endings (Lines 9–10). Unlike the FC variants, VTHarm and DAT-VAE do not have V scores because they do not produce chords with actual pitch values. Instead, VTHarm produces chord labels in root position while DAT-VAE produces a set of pitch classes without chord position information.

FC– consistently achieves the highest number of inverted chords IC because it inverts chord positions to minimize voice fluctuation (Line 8, 16). Unlike FC–, FC needs to satisfy the cadence constraint that fixes chords in root position at phrase or section endings (Lines 9–10). Hence, its IC scores are lower than those of FC–. Since VTHarm and DAT-VAE do not produce chords with chord position information, they do not have IC scores.

FC– consistently achieves low chord histogram entropy H for all test sets, indicating its narrow range of chords used. It chooses the most frequently used chords by maximizing chord transition probability (Lines 3, 14), thus restricting its

choices of chords. In contrast, by catering to cadence, FC can select chords that do not necessarily have the largest transition probabilities. Thus, it has a slightly higher H compared to FC-. Unlike the FC variants, both DAT-VAE and VTHarm do not explicitly maximize chord transition probability. Instead, they minimize chord reproduction error and KL divergence between the learned and target distributions. Their H scores appear to depend on the training and testing sets. In particular, VTHarm achieves the lowest H score for Pop-S but the highest H score for Pop-L and Jazz-L. On the other hand, DAT-VAE's H score is similar to that FC- for Pop-S and Pop-L, but similar to that of VTHarm for Jazz-L.

FC and FC- have very similar dissonance D scores. Both FC and FC- balance between the maximization of chord transition probabilities (Lines 3, 14) and the minimization of dissonance (Lines 4, 15). DAT-VAE has the lowest D because it does not need to handle chord transition probability. By minimizing chord reproduction error and KL divergence of chord distributions, it could choose chords with the lowest dissonance. In contrast, VTHarm has a self-attention mechanism that allows it to choose less frequently-used chords. Being less frequently-used, these chords tend to have large dissonance. Therefore, VTHarm has the largest dissonance D compared to all other methods.

Chord-tone to non-chord tone ratio R0 is a measure that is negatively correlated to dissonance D because more chord tones leads to lower dissonance. The FC variants have similar large R0 for their small D. In contrast, the R0 scores of VTHarm and DAT-VAE seem to be positively, rather than negatively, correlated to their D scores, which is counter-intuitive. In particular, DAT-VAE has the lowest D and lowest R0 among all methods, whereas VTHarm has the highest D and relatively high R0.

R1 is expected to be larger than (or equal to) R0 because it counts proper non-chord tones as chord tones. All methods exhibit this property. Thus, R0 could replace R1 as a better performance metric because it does not require proper non-chord tone, which is a concept that is not defined in music theory.

Overall, FC achieves the best performance compared to other methods. It caters to cadence by sacrificing chord diversity slightly (slightly small H). It chooses chords with smooth voices (small V) and relatively low dissonance (small D), indicating good chord-melody matching. Although deep neural networks may excel in an

individual metric, they do not handle cadence and voice fluctuation. Moreover, their D and R0 scores have a relationship that is opposite to the expected negative correlation. There is noticeable difference when FC is applied with Pop-S compared to Pop-L, but the difference is not significant. This indicates that FC’s performance is stable even with a small reference set.

4.4.3 Qualitative Evaluation

(a) Qualitative Comparison of Various Methods

Figure 4.2 illustrates the chords generated by various methods for the first 8 bars of *Colors of the Wind*, a song in the testing set. Note that VTHarm and DAT-VAE generate chord pitches without information of exact chord positions. For display purpose, their chord labels are inferred from the generated pitches and their chords are shown in root position. The original chords of *Colors of the Wind* are included to serve as a reference for evaluation.

The figure reveals that FC and FC– produce chords that are stylistically similar to the original chords of *Colors of the Wind*. They include F and C, the tonic and dominant chords, that form the harmonic foundation for the melody, which is in the key of F major. Furthermore, they include the relative minor chord Dm, which is integral to the original harmonization. Dm shares two pitch classes, namely F and A, with the F chord. This commonality ensures smooth transitions between major and minor chords within the key.

Yet, FC and FC– remain creative in how the same chords are differently applied to phrase and section endings indicated by ‘P’ and ‘S’. For example, both FC and FC– reproduce the original Dm chord in bar 8. However, in bar 4, FC uses Dm, while FC– uses F to conclude the phrase. Although both FC and FC– do not use the original Am chord in bar 4, their chord choice remains harmonious with the melody.

Figures 4.2 also illustrates how FC handles cadence. For instance, FC produces the chord transition Dm–Dm in bar 8, which is identical to the original chords. Its bass notes descend relatively smoothly. In contrast, FC– is unable to consistently produce cadences because it does not handle cadence during chord generation. For example, in bar 4, it produces Dm–F chord transition, which is not a proper cadence.

The figure displays a musical score for the first 8 bars of "Colors of the Wind". It consists of six staves: Melody, Original, VTHarm, DAT-VAE, FC-, and FC. The Melody staff is in 4/4 time with a key signature of one flat (Bb). The Original staff shows the original chord progression. The VTHarm staff shows a different progression. The DAT-VAE staff shows a progression with Csus4 and G7. The FC- and FC staves show a progression with F/C and Gm/D. The figure is labeled with 'P' and 'S' above the Melody staff.

Staff	Bar 1	Bar 2	Bar 3	Bar 4	Bar 5	Bar 6	Bar 7	Bar 8
Melody	1	2	3	4	5	6	7	8
Original	F	Dm	F	Am	Dm	Bb	Gm7 C7	Dm
VTHarm	Bbmaj7	Gm Bb	F C	Dm	Bbmaj7	Gm Bb	F	Dm
DAT-VAE	F	Dm	F	Dm	Csus4 G7	C	Csus4 Dm	
FC-	F C/E	Gm/D	F/C C	Dm F/C	C	Gm/Bb Bb	C F/C	Dm
FC	F C/E	Gm/D	F/C C	Dm	F/C C	Gm/Bb Bb	C F/C	Dm

Figure 4.2: Chords generated by various methods for the first 8 bars of *Colors of the Wind*.

VTHarm produces a chord progression that noticeably deviates from the original chord progression. It produces Bbmaj7 twice with unexpected chord progressions that are not found in the original chord sequence. In bars 1 and 2, the chords change from Bbmaj7 to Gm to Bb, resulting in a harmonic shift that may sound unusual compared. Moreover, the tonic F chord found in bar 1 of the original chord sequence is absent in the chord sequence of VTHarm. In fact, without the melody, it is hard to qualitatively determine the key of VTHarm's chord progression.

Similarly, DAT-VAE introduces surprising progressions that include Csus4 and G7. Nevertheless, it fares slightly better than VTHarm since it reproduces the tonic F chord in bar 1 that reinforces the melody's key. However, the sequences of Csus4 – G7 – C and Csus4 – D are disruptive to the overall chord progression. The Csus4 chord introduces tension and creates a sense of expectation for resolution. In the Csus4 – G7 – C progression, although Csus4 resolves to C, G7 introduces an additional layer of complexity and shifts the harmonic focus away from the melody key. Similarly, in the Csus4 – D progression, Csus4 creates a suspension that disrupts

the smooth flow of the progression and leads to an unexpected harmonic shift when it moves to D.

These examples show that the harmonization results of VTHarm and DAT-VAE can diverge from the intended pop style and melody key. This could be due to their training on a diverse collection of pop music, diluting the style specificity of the generated chord sequence. In other words, their results may not be harmonically coherent. This disparity underscores the complexity of capturing the nuances of pop harmonization.

4.4.4 Subject Evaluation

Subject evaluation was conducted to compare the perceived harmonization quality of various methods. 27 subjects with strong music background (play piano, at least 5 years of music training) were recruited via CloudConnect. Each subject was presented with two 8-bar songs with chord accompaniment. One of the pair was either the original manual harmonization or FlexChord’s result. The other was a result of either VTHarm or DAT-VAE. The subject was asked to pick the one that was more harmonious. In addition, the subject was asked to rate how harmonious each song was on a 5-point Likert scale. The ratings were used to verify whether the subject’s choice was consistent with the ratings. Each subject repeated the above procedure for 40 pairs of songs.

The subjects’ assessments were grouped into one of the following cases:

- O+: original harmonization is more harmonious than DNN’s result.
- O−: original harmonization is less harmonious than DNN’s result.
- F+: FlexChord’s result is more harmonious than DNN’s result.
- F−: FlexChord’s result is less harmonious than DNN’s result.

Table 4.4 tabulates the subjects’ assessments. A large majority of subjects judge the original harmonizations as more harmonious than those of VTHarm and DAT-VAE (68.3% and 74.7% respectively). This expected result indicates that the subjects understand and appreciate harmonization. Among them, DAT-VAE’s results are the least harmonious. Similarly, most subjects judge FlexChord’s harmonizations

Table 4.4: Subject evaluation results.

Cases	VTHarm	DAT-VAE
O+	179 (68.3%)	195 (74.7%)
O−	83 (31.7%)	66 (25.3%)
F+	158 (60.3%)	159 (60.9%)
F−	104 (39.7%)	102 (39.1%)
O+ and F+	124 (47.3%)	132 (50.6%)
O+ and F−	55 (21.0%)	63 (24.1%)
O− and F+	34 (13.0%)	27 (10.3%)
O− and F−	49 (18.7%)	39 (15.0%)
Total votes	262 (100%)	261 (100%)

to be more harmonious than those of VTHarm and DAT-VAE (60.3% and 60.9% respectively), with DAT-VAE results being the least harmonious. In addition, more subjects judge both the original and FlexChord’s harmonizations to be more harmonious than those of the DNNs (47.3%, 50.6%), compared to the other three cases. These subject test results indicate that FlexChord’s harmonizations are better than those of the DNNs, and are comparable to those of the original harmonizations.

4.4.5 Flexible Harmonization Style

(a) Harmonization with Various Styles

FlexChord (FC) can generate various harmonization styles for the same melody by using chord transition models of the corresponding styles. To test FC’s flexibility beyond pop style, the alternative style should be distinct enough to highlight differences in chord types and demonstrate style-specific characteristics. One such style is jazz. Jazz features more chromatic movement, where notes shift by small intervals. It also embraces a wider range of dissonance and chords that extend from simple triads compared to pop, which tends to rely on straightforward harmonic progressions and cadences.

To evaluate FC’s ability to generate quality jazz harmonizations without training, two tests were conducted using a chord transition model derived from Jazz-L examples. The first test involved generating a jazz harmonization for the jazz song *Israel*. Figure 4.3 illustrates the jazz chords generated by FlexChord for bars 17–24

Figure 4.3 displays the original and jazz chords generated by FC for bars 17 to 24 of the song *Israel*. The score is presented in three staves: Melody, Original, and FC. The key signature is three sharps (F#, C#, G#) and the time signature is 4/4. The FC staff provides jazz chord progressions for the original chords. A blue 'P' is above bar 20 and a blue 'S' is above bar 24.

Bar	Melody	Original	FC
17		C#m7	C#m7
18		F#/C#	F#/C#
19		G#m7	B/D# G#7/D#
20		G#m7	G#m/D# G#m
21		G# C#	A#m7/G# G#m7
22		F# D#7	C#m7/G# F#/A#
23		G#m7 Fm7(b5)	G#m7 Fm7(b5)
24		A#m7(b5) D#7	BMaj7/A# G#m7

Figure 4.3: The original and jazz chords generated by FC for bars 17 to 24 of *Israel*.

of the song. It shows how FC's chord progression includes chromatic movements that add harmonic richness and tension. It also employs the classic ii7–V–I(maj7) progression (C#m7–F#–B(maj7) in bars 18–19, 22–24), which is a hallmark of jazz. This progression creates a strong sense of anticipation and resolution, with the ii7 chord leading smoothly to the V chord before resolving to the I chord. Moreover, FC introduces chromatic transitions to chords like VII, vii and vi (G#, G#m and A#m in bars 19–21) or their seventh extensions. These transitions, achieved by minimizing voice fluctuation, demonstrate FC's ability to produce style-specific jazz harmonies while maintaining harmonic coherence with a jazz melody.

In the second test, FC's flexibility was evaluated by generating both pop and jazz harmonizations for the same pop melody. Figure 4.4 shows the chords generated by FC for the first 8 bars of *Colors of the Wind* using either the pop or jazz chord transition model. The pop version features basic triads like I, ii, V and vi (F, Gm, C, Dm), while the jazz version enhances these with seventh chord extensions, such as ii7, II7, iii7 and vi7 (Gm7, G7, Dm7, Am7). The jazz version also features chromatic voice leading in the transitions between chords. For example in bar 2 to 3, the chord progression moves from G7/F to Gm/F and then Am7/E. This sequence creates

Figure 4.4: Pop and jazz chords generated by FC for the first 8 bars of *Colors of the Wind*.

smooth chromatic movement in the form of small pitch change of F to E in the bass and B to B $\flat$ to A in the inner voice.

Another possibility is the harmonization of a jazz melody with pop-style chords. While FC is capable of this, the results are not discussed since pop chord progressions are typically less rich than jazz chord progressions, which support the jazz melody better.

(b) Custom Harmonization Style

FlexChord can generate harmonizations in a custom style using as little as a single example music piece. This is achieved by creating a customized chord transition model, where chord transitions and probabilities are manually defined based on the example piece. This approach enables FlexChord to produce custom style harmonizations without the need for additional training.

To test FlexChord’s ability to harmonize in a custom style, Hitchcock’s *Psycho* was selected as the reference music for generating spooky style. Chord pairs were extracted from *Psycho* and chord transition probabilities were computed for the spooky chord transition model.

Figure 4.5 illustrates the result of harmonizing the first 8 bars of *In the Hall of the Mountain King* with the spooky chord transition model. Compared to the original harmonization, FlexChord generates a higher concentration of dissonant chords, similar to style of *Psycho*. For instance, the A $\sharp$ dim and F $\sharp$ 7 chords in bars 2 and 6 clash with the melody’s natural notes, intensifying the overall tension. In

Figure 4.5: Custom-style chords for the first 8 bars of *Hall of the Mountain King*.

contrast, the original chords are triads such as Bm, C#, and D which are more consonant with the melody.

4.4.6 Full-Length Test

(a) Test Procedure

The objective of this test is to assess FlexChord’s ability to generate harmonization across extended musical structures. Both FC and FC– were executed on the full song of *Colors of the Wind* that includes diverse melodic variations and intervals. For comparison, VTHarm and DAT-VAE were executed on consecutive 8-bar phrases of *Colors of the Wind* after being trained on the Pop-L training set.

(b) Qualitative Results and Discussions

Figure 4.6 illustrates selected fragments of the test results as it is too long to show all 62 bars of *Colors of the Wind*. For all methods, the results for the first 8 bars are identical to those of the short-phrase test (Fig. 4.2). With the exception of bars 15 and 16, the melody of section **A** is repeated exactly in section **B**. For FC, the chords of **A** are also repeated exactly in **B**, due to the enforcement of repetition constraint (Algorithm 1, Lines 11, 12). Both section **A** and **B** constitute the verse of the song. Slight variations of this verse can also be found in sections **D**, **E** and **G**.

The chord transitions to section **C** (bars 16 to 17) of FC and FC– create a relevant major-major contrast from chord F to Am. Both chords share two pitches, which are A and C. Similarly, VTHarm creates a major-minor contrast with chord Dm transitioning to Bb. Both chords also share two pitches, which are D and F.

Fragment A (Measures 1-4, 8)

VTHarm:	B♭Maj7	∕	Gm	B♭	F	C	Dm	∕	Dm	∕
DAT-VAE:	F	∕	Dm	∕	F	∕	Dm	∕	∕	∕
FC-:	F	C/E	Gm/D	∕	F/C	C	Dm F/C	∕	∕	∕
FC:	F	C/E	Gm/D	∕	F/C	C	Dm	∕	Dm	∕

Fragment B (Measures 9-10, 16-18)

VTHarm:	∕	∕	C	∕	F	∕	B♭	∕	Am	∕
DAT-VAE:	F	∕	Dm	∕	Dm	∕	∕	∕	Gm	B♭add#4
FC-:	F/C	C	Gm/B♭ F/A	∕	F/A	∕	Am	Dm7	∕	Gm7/D
FC:	F	C/E	Gm/D F/A	∕	F/A	F	Am	Dm7	F	Dm

Fragment C (Measures 19-29, 43)

VTHarm:	Asus4	∕	F	∕	B♭	∕	F	C	∕	∕	F	∕
DAT-VAE:	Gm7	F	Dm	Am	Csus#4 F	Dm	F	Dm	∕	∕	F/A	∕
FC-:	F/C	∕	F/C	∕	∕	C	F/C	C	Gm/B♭	∕	∕	F
FC:	F/C	∕	F	∕	∕	C/E	F	C/E	Gm/D	∕	∕	B♭

Fragment D (Measures 44-45, 46-47, 48-49, 61-62)

VTHarm:	F	Am	Dm	∕	Gm	∕	B♭	∕	F	∕	G7	C7	F	∕
DAT-VAE:	F+	E♭	F	∕	E♭+	Gm	F	C	F	CMaj7	G7	∕	∕	F
FC-:	C	∕	∕	B♭/D	C	Gm/B♭	F/A	C	F/C	C	C	B♭/D	F/C	∕
FC:	C/E	∕	F	B♭	C	Gm/B♭	Dm/A Gm	F	C/E	∕	C	B♭/D	F	∕

Figure 4.6: Selected fragments from the test results of the full song of *Colors of the Wind*.

However, DAT-VAE does not create contrast when transitioning to **C**. Instead, it produces repeated chords of Dm–Dm.

Section **C** also contains the chorus of the song, with the melodic climax located in bar 17 and its partial resolution in bar 18. The tension leading to the climax is built by the ascending pitches F–A–C in bar 16, which leap to F in the next octave, followed by descending notes into bar 18. FC follows the same trend by selecting chords that initially create dissonance with the melody, followed by chords that are consonant with the melody. For example, in bar 17, the pitch E of chord Am clashes

with the melody pitches F and D, producing a high dissonance. This is followed by a F–Dm cadence in bar 18, which is consonant with the melody. Thus, FC effectively builds tension to the climax and then resolves it. The results of VTHarm follow the same trend but uses the chord transition B♭–Am instead to build and release tension.

In contrast, the results of FC– do not follow the same trend. While FC– builds tension to the climax using the same chords as FC in bar 17, it does not release tension due to its lack of handling cadence in bar 18. At bar 18, it repeats the Dm7 chord and transitions to another seventh chord (Gm7). The results of DAT-VAE go against the melodic trend. It repeats the Dm chord in bar 17 which is consonant with the melody transitions to Gm–B♭add♯4 that are dissonance with the melody.

The chord transitions to **F** (bars 43–44) of FC and FC– create a tonic-dominant contrast with the transition from F to C. In contrast, VTHarm does not feature any chord contrast in the transition from F to F. DAT-VAE generates an F to F+ transition, which introduces a chromatic shift. This transition is harmonically incoherent with the key of F major and the pop style because F+ chord creates an unexpected dissonance.

Colors of the Wind ends in bar 62 with the F note. Interestingly, all methods generate the F chord or a chord rooted in F. In particular, FC, FC– and VTHarm generate an F to F chord transition. On the other hand, DAT-VAE generates the chord G7 which contains the pitch F, followed by chord F. This shows a general agreement among the methods that F is an important note to include in the chord.

4.4.7 Summary

Quantitative results illustrate the strengths of FlexChord for short phrases. It directly minimizes voice fluctuation to produce chord progression with smooth voice leading (small V). It excels in generating a large number of inverted chords (large IC) and caters to cadence by sacrificing chord diversity (small H) slightly. It chooses chords with relatively low dissonance (small D) and achieves a favorable ratio of chord tones to non-chord tones (large R0), indicating good chord-melody matching.

Qualitative results show that FlexChord achieves style specificity by producing chord sequence that matches the style of a selected set of example music. It produces

a pop chord sequence that is similar to the original chord sequence, but remains creative in how the same chords are used as cadences to mark phrase and section endings. Although FlexChord does not always reproduce the original cadence, the alternate cadence that FlexChord selects remains harmonious with the melody.

Subject test results reveal that most subjects judge FlexChord’s harmonizations to be more harmonious than DNN harmonizations. In addition, more subjects judge both the original and FlexChord’s harmonizations to be more harmonious than those of the DNNs, compared to the other cases. This indicates that FlexChord’s harmonizations are better than those of the DNNs, and are comparable to those of the original harmonizations.

Additional tests results show that FlexChord is flexible at various style levels. Custom or specific styles are explored using a chord transition model of the required style, without the need for training on large datasets. Tests using a jazz chord transition model show that FlexChord can generate high-quality jazz chord sequences that maintain harmonic coherence with the melody. In a separate test, FlexChord uses a custom chord transition model derived from a single example piece. It successfully generates a custom style harmonization that incorporates more dissonant chords than the original, effectively conveying tension.

Full-length melody test shows that FlexChord can produce harmonically coherent chord progressions that match a full-length melody, complete with global music structure. Chord repetitions and key-relevant chords are used to ensure consistency among chords and effective matching with the melody. In addition, FlexChord generates cadences that result in the tension and resolution needed for contrasting sections. Overall, FlexChord shows promise as a flexible harmonizer that can create pop and jazz chord sequences for complete songs.

Chapter 5

Remaining Work and Conclusion

Automated music composition (AMC) is an important topic that consists of two main tasks: (1) the generation of music structure and melody, and (2) the generation of chord and accompaniment. Since melody generation is well studied, this thesis focuses on the second task of generating chords and accompaniment.

This thesis proposes a method of harmonization called FlexChord that overcomes the shortcomings of existing methods. FlexChord achieves style specificity by producing chord progressions that match the style of a selected set of example music. It applies non-iterative, multi-objective optimization during chord generation that does not require training. Thus, it can easily produce chord progression of a specific style given the corresponding example set. The requirement of cleaning, annotating and training on large datasets is avoided.

Test results show that FlexChord can produce a large number of inverted chords that result in smooth voice leading. It caters to cadence by sacrificing chord diversity slightly and choosing chords with relatively low dissonance. Thus, it achieves a favorable ratio of chord tones to non-chord tones, which indicates good chord-melody matching. On the other hand, deep neural networks for harmonization such as VTHarm and DAT-VAE score higher chord histogram entropy but lower chord tone to non-chord tone ratio. While they may excel in an individual metric, they do not handle cadence and voice leading.

Being trained on a diverse collection of pop music, VTHarm and DAT-VAE show evidence of style dilution and harmonic incoherence with the melody’s key. In contrast, FlexChord generates chord sequence that matches the style of the reference chord transition model. FlexChord’s chord sequence maintains harmonic coherence with the melody’s key and trends towards the melody climax. It provides stylistically

appropriate tension and resolution that support the melody. FlexChord takes care of global music structure such as section repetitions and produces chords sequences that match music structure.

FlexChord can harmonize the same melody with various styles by using chord transition models of the corresponding styles. Furthermore, it easily produces a chord sequence of a custom style with as little as one example music piece to convey more tension than the original chord sequence. Subject test results show that FlexChord’s harmonization is better than those of comparable DNNs and is close to that of human harmonization.

In summary, FlexChord shows promise as a flexible harmonizer that can generate style-specific chord progressions at various style levels for complete songs. The chords generated by FlexChord can be utilized for generating style-specific accompaniments to full-length melodies, which is the remaining work of this thesis. Generation of accompaniment involves transforming the generated chords into multiple rhythmic voices that fit a specific accompaniment style. For instance, chords with tightly packed notes can be “unrolled” by spreading them over a wider pitch range and cycling through them. Additionally, the original note durations can be adjusted to fit the rhythmic patterns of the melody. The generated accompaniment should also match the global music structure of a full-length melody. This includes producing harmonic and rhythmic contrast between different sections, such as verse and chorus, while ensuring rhythmic consistency within each section. These parts will be developed to complete the thesis.

References

- [Agu+10] G. Aguilera et al. “Automated generation of contrapuntal musical compositions using probabilistic logic in Derive”. In: *Mathematics and Computers in Simulation* 80.6 (2010), pp. 1200–1211.
- [AW04] Moray Allan and Christopher Williams. “Harmonising chorales by probabilistic inference”. In: *Advances in neural information processing systems* 17 (2004).
- [AM09] Torsten Anders and Eduardo R Miranda. “A computational model that generalises Schoenberg’s guidelines for favourable chord progressions”. In: *Sound and Music Computing (SMC)*. 2009, pp. 48–52.
- [And21] Anderson, Chris and Carlton, Dave and Miyakawa, Ryan and Schwachhofer, Dennis. *Hooktheory*. <https://www.hooktheory.com>. Accessed: 2022-10-06. 2021.
- [BC15] L Bigo and D Conklin. “A viewpoint approach to symbolic music transformation”. In: *International Symposium on Computer Music Multidisciplinary Research*. Springer, 2015, pp. 213–227.
- [BHP20] Jean-Pierre Briot, Gaëtan Hadjeres, and François-David Pachet. *Deep learning techniques for music generation*. Vol. 1. Springer, 2020.
- [BP20] Jean-Pierre Briot and François Pachet. “Deep learning for music generation: challenges and directions”. In: *Neural Computing and Applications* 32.4 (2020), pp. 981–993.
- [BLB07] J. Buchholz, E. Lee, and J. Borchers. *coJIVE: A System to Support Collaborative Jazz Improvisation*. Tech. rep. RWTH Aachen Univ, 2007. URL: <https://api.semanticscholar.org/CorpusID:16092297>.

- [Cap04] William E Caplin. “The classical cadence: Conceptions and misconceptions”. In: *Journal of American Musicological Society* 57.1 (2004), pp. 51–118.
- [CLC13] Pei-Chun Chen, Keng-Sheng Lin, and Homer H. Chen. “Automatic accompaniment generation to evoke specific emotion”. In: *IEEE International Conference on Multimedia and Expo (ICME)*. 2013, pp. 1–6. DOI: [10.1109/ICME.2013.6607423](https://doi.org/10.1109/ICME.2013.6607423).
- [Che+21] Yi-Wei Chen et al. “SurpriseNet: Melody Harmonization Conditioning on User-controlled Surprise Contours”. In: *International Society for Music Information Retrieval Conference (ISMIR)*. 2021, pp. 105–112.
- [CC11a] C.-H. Chuan and E. Chew. “Generating and evaluating musical harmonizations that emulate style”. In: *Computer Music Journal* 35.4 (2011), pp. 64–82.
- [CC11b] Ching-Hua Chuan and Elaine Chew. “Generating and evaluating musical harmonizations that emulate style”. In: *Computer Music Journal* 35.4 (2011), pp. 64–82.
- [Civ+22] Miguel Civit et al. “A systematic review of artificial intelligence-based music generation: Scope, applications, and future trends”. In: *Expert Systems with Applications* (2022), p. 118190.
- [CL17] Tom Collins and Robin Laney. “Computer-generated stylistic compositions with long-term repetitive and phrasal structure”. In: *Journal of Creative Music Systems* 1.2 (2017).
- [Col+11] Tom Collins et al. “Modeling pattern importance in Chopin’s mazurkas”. In: *Music Perception* 28.4 (2011), pp. 387–414.
- [CSG17] Florian Colombo, Alexander Seeholzer, and Wulfram Gerstner. “Deep artificial composer: A creative neural network model for automated melody generation”. In: *Computational Intelligence in Music, Sound, Art and Design: 6th International Conference, EvoMUSART 2017, Amsterdam, The Netherlands, April 19–21, 2017, Proceedings 6*. Springer. 2017, pp. 81–96.

- [Con18] Darrell Conklin. “Chord sequence generation with semiotic patterns”. In: *Machine Learning and Music Generation*. Routledge, 2018, pp. 6–20.
- [CSH18] Nailson Cunha, Anand Subramanian, and Dorien Herremans. “Generating guitar solos by integer programming”. In: *Journal of the Operational Research Society* 69 (2018), pp. 971–985.
- [Dai+21] Shuqi Dai et al. “Controllable deep melody generation via hierarchical music structure representation”. In: (2021), pp. 143–150.
- [DPZZ20] R. De Prisco, G. Zaccagnino, and R. Zaccagnino. “EvoComposer: An Evolutionary Algorithm for 4-Voice Music Compositions”. In: *Evolutionary Computation* 28.3 (Sept. 2020), pp. 489–530. ISSN: 1063-6560. DOI: [10.1162/evco_a_00265](https://doi.org/10.1162/evco_a_00265).
- [Dég+22] Ken Déguernel et al. “Personalizing AI for co-creative music composition from melody to structure”. In: *Sound and Music Computing (SMC)*. 2022, pp. 314–321.
- [Dos05] Martin Dostál. “Genetic Algorithms as a Model of Musical Creativity - on Generating of a Human-Like Rhythmic Accompaniment”. In: *Comput. Artif. Intell.* 24.3 (2005), pp. 321–340.
- [Ebc88a] K. Ebcioglu. “An expert system for harmonizing four-part chorales”. In: *Computer Music Journal* 12.3 (1988), pp. 43–51.
- [Ebc88b] Kemal Ebcioglu. “An expert system for harmonizing four-part chorales”. In: *Computer Music Journal* 12.3 (1988), pp. 43–51.
- [Ehr05] Matthias Ehrgott. *Multicriteria optimization*. Vol. 491. Springer Science & Business Media, 2005.
- [FG11] Alan Freitas and Frederico Guimaraes. “Melody harmonization in evolutionary music using multiobjective genetic algorithms”. In: *Sound and Music Computing (SMC)*. 2011.
- [FM81] J. Fux and A. Mann. *Study of Counterpoint: From Johann Joseph Fux’s Gradus ad Parnassum*. WW Norton, 1981.
- [GJ02] A. Gartland-Jones. “Can a genetic algorithm think like a composer?”. In: *Generative Art Conference*. 2002.

- [GM07] Michael Geis and Martin Middendorf. “An ant colony optimizer for melody creation with baroque harmony”. In: *IEEE Congress on Evolutionary Computation*. IEEE. 2007, pp. 461–468.
- [GC07] E. Gilbert and D. Conklin. “A probabilistic context-free grammar for melodic reduction”. In: *Proceedings of International Workshop on Artificial Intelligence and Music at International Joint Conference on Artificial Intelligence*. Hyderabad, India, 2007.
- [Gog06] M. Gogins. “Score generation in voice-leading and chord spaces”. In: *International Computer Music Conference, ICMC*. 2006.
- [Goi+19] Izaro Goienetxea et al. “Statistics-based music generation approach considering both rhythm and melody coherence”. In: *IEEE Access* 7 (2019), pp. 183365–183382.
- [Got+21] Mark Gotham et al. *OPEN MUSIC THEORY*. en. 2021. URL: <https://viva.pressbooks.pub/openmusictheory/>.
- [GN21] T. Greer and S. Narayanan. “Harmonize this Melody: Automatic Four-Part Harmony Generation using Neo-Riemannian Voice-Leading”. In: *International Society for Music Information Retrieval Conference (ISMIR) ISMIR, Late-Breaking Demo*. 2021.
- [GDG21] Jacek Grekow and Teodora Dimitrova-Grekow. “Monophonic Music Generation With a Given Emotion Using Conditional Variational Autoencoder”. In: *IEEE Access* 9 (2021), pp. 129088–129101. DOI: [10.1109/ACCESS.2021.3113829](https://doi.org/10.1109/ACCESS.2021.3113829).
- [GD97] Lorin Grubb and Roger B. Dannenberg. “A Stochastic Method of Tracking a Vocal Performer”. In: *International Computer Music Conference, ICMC*. Michigan Publishing, 1997.
- [Gwe02] N. Gwee. “Complexity and Heuristics in Rule-Based Algorithmic Music Composition”. PhD thesis. Louisiana State University, 2002.
- [HPN17] Gaëtan Hadjeres, François Pachet, and Frank Nielsen. “DeepBach: a steerable model for Bach chorales generation”. In: *International Conference on Machine Learning*. 2017, pp. 1362–1371.

- [HBE20] Shunit Haviv Hakimi, Nadav Bhonker, and Ran El-Yaniv. “BebopNet: Deep Neural Models for Personalized Jazz Improvisations”. In: *International Society for Music Information Retrieval Conference (ISMIR)*. 2020, pp. 828–836.
- [HCC17] D. Herremans, C.-H. Chuan, and E. Chew. “A Functional Taxonomy of Music Generation Systems”. In: *ACM Computing Surveys* 50 (2017), pp. 69–101. DOI: **10.1145/3108242**.
- [HS13] Dorien Herremans and Kenneth Sörensen. “FuX, an Android app that generates counterpoint”. In: *IEEE Symposium on Computational Intelligence for Creativity and Affective Computing, CICAC 2013*. IEEE, 2013, pp. 48–55. DOI: **10.1109/CICAC.2013.6595220**.
- [Hie19] Hiehn, Steve. *Chord Melody Dataset*. <https://github.com/shiehn/chord-melody-dataset>. Accessed: 2022-10-06. 2019.
- [HFM91] Hermann Hild, Johannes Feulner, and Wolfram Menzel. “HARMONET: A neural net for harmonizing chorales in the style of JS Bach”. In: *Advances in neural information processing systems* 4 (1991).
- [Hsi+21] Wen-Yi Hsiao et al. “Compound Word Transformer: Learning to Compose Full-Song Music over Dynamic Directed Hypergraphs”. In: *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021*. AAAI Press, 2021, pp. 178–186. DOI: **10.1609/AAAI.V35I1.16091**.
- [HC05] C. Z. A. Huang and E. Chew. “Palestrina Pal: a grammar checker for music compositions in the style of Palestrina”. In: *Proceedings of the Conference on Understanding and Creating Music*. 2005.
- [Hua+17] Cheng-Zhi Anna Huang et al. “Counterpoint by Convolution”. In: *International Society for Music Information Retrieval Conference (ISMIR)*. 2017, pp. 211–218.
- [Hua+18] Cheng-Zhi Anna Huang et al. “Music Transformer: Generating Music with Long-Term Structure”. In: *International Conference on Learning Representations*. 2018.

- [Hua+19] Cheng-Zhi Anna Huang et al. “Music Transformer: Generating Music with Long-Term Structure”. In: *International Conference on Learning Representations, ICLR*. OpenReview.net, 2019.
- [HH20] Chih-Fang Huang and Cheng-Yuan Huang. “Emotion-based AI music generation system with CVAE-GAN”. In: *IEEE Eurasia Conference on IOT, Communication and Engineering (ECICE)*. IEEE. 2020, pp. 220–222.
- [Hua+20] Tsung-Min Huang et al. “Play it again IMuCo! music composition to match your mood”. In: *International Conference on Transdisciplinary AI (TransAI)*. IEEE. 2020, pp. 9–16.
- [Hur94] D. Huron. “Interval-class content in equally tempered pitch-class sets: Common scales exhibit optimum tonal consonance”. In: *Music Perception* 11.3 (1994), pp. 289–305.
- [Hur01] D. Huron. “Tone and voice: A derivation of the rules of voice-leading from perceptual principles”. In: *Music Perception* 19.1 (2001), pp. 1–64.
- [Hur16] D. Huron. *Voice Leading: The Science Behind a Musical Art*. MIT press, 2016.
- [Jaq+16] Natasha Jaques et al. “Generating Music by Fine-Tuning Recurrent Neural Networks with Reinforcement Learning”. In: *Deep Reinforcement Learning Workshop, NIPS*. 2016.
- [JKA17] Jaehun Jeong, Yuseung Kim, and Chang Wook Ahn. “A multi-objective evolutionary approach to automatic melody generation”. In: *Expert Systems with Applications* 90 (2017), pp. 50–61.
- [JYL24] Shulei Ji, Xinyu Yang, and Jing Luo. “A Survey on Deep Learning for Symbolic Music Generation: Representations, Algorithms, Evaluations, and Challenges”. In: *ACM Comput. Surv.* 56.1 (2024), 7:1–7:39. DOI: **10.1145/3597493**.
- [Ji+23] Shulei Ji et al. “RL-Chord: CLSTM-Based Melody Harmonization Using Deep Reinforcement Learning”. In: *IEEE Transactions on Neural Networks and Learning Systems* (2023), pp. 1–14. DOI: **10.1109/TNNLS.2023.3248793**.

- [JKW17] Daniel D Johnson, Robert M Keller, and Nicholas Weintraut. “Learning to create jazz melodies using a product of experts.” In: *International Conference on Innovative Computing and Cloud Computing*. 2017, pp. 151–158.
- [KP+23] Maximos Kaliakatsos-Papakostas et al. “An HMM-Based Approach for Cross-Harmonization of Jazz Standards”. In: *Applied Sciences* 13.3 (2023). ISSN: 2076-3417. DOI: [10.3390/app13031338](https://doi.org/10.3390/app13031338).
- [KGR18] Tetsuro Kitahara, Sergio Giraldo, and Rafael Ramírez. “JamSketch: Improvisation Support System with GA-Based Melody Creation from User’s Drawing”. In: *Music Technology with Swing*. Cham: Springer International Publishing, 2018, pp. 509–521. ISBN: 978-3-030-01692-0.
- [Kle05] J. A. Klein. “Pattern-Based Software Framework for Computer Aided Jazz Improvisation”. PhD thesis. RWTH Aachen, 2005.
- [KK23] Peter van Kranenburg and Eoin Kearns. “Algorithmic Harmonization of Tonal Melodies Using Weighted Pitch Context Vectors”. In: *International Society for Music Information Retrieval Conference (ISMIR)*. 2023, pp. 391–397.
- [LJ04] Hong-Ru Lee and J-SR Jang. “i-Ring: A system for humming transcription and chord generation”. In: *IEEE International Conference on Multimedia and Expo (ICME)*. Vol. 2. IEEE. 2004, pp. 1031–1034.
- [Leo24] W. K. Leow. *Foundation of Media Computing*. Third Draft (to be published). 2024.
- [Lia16] Feynman Liang. “Bachbot: Automatic composition in the style of Bach chorales”. MA thesis. University of Cambridge, 2016.
- [LRL17] Hyungui Lim, Seungyeon Rhyu, and Kyogu Lee. “Chord Generation from Symbolic Melody Using BLSTM Networks”. In: *International Society for Music Information Retrieval Conference (ISMIR)*. 2017, pp. 621–627.
- [LZ22] Mengqian Lin and Rui Zhao. “A Study of Piano-Assisted Automated Accompaniment System Based on Heuristic Dynamic Planning”. In: *Computational Intelligence and Neuroscience 2022* (2022).

- [LT12] Chien-Hung Liu and Chuan-Kang Ting. “Polyphonic accompaniment using genetic algorithm with music theory”. In: *IEEE Congress on Evolutionary Computation*. 2012, pp. 1–7. DOI: [10.1109/CEC.2012.6252869](https://doi.org/10.1109/CEC.2012.6252869).
- [LWY18] Hao-Min Liu, Meng-Hsuan Wu, and Yi-Hsuan Yang. “Lead sheet generation and arrangement via a hybrid generative model”. In: *International Society for Music Information Retrieval Conference (ISMIR)*. 2018, pp. 23–27.
- [LY18] Hao-Min Liu and Yi-Hsuan Yang. “Lead Sheet Generation and Arrangement by Conditional Generative Adversarial Network”. In: *IEEE International Conference on Machine Learning and Applications, ICMLA*. IEEE, 2018, pp. 722–727. DOI: [10.1109/ICMLA.2018.00114](https://doi.org/10.1109/ICMLA.2018.00114).
- [Lop+17] Henrique Barros Lopes et al. “Combining rules and proportions: A multiobjective approach to algorithmic composition”. In: *IEEE Congress on Evolutionary Computation (CEC)*. IEEE. 2017, pp. 2282–2289.
- [MKS16] D. Makris, I. Kayrdis, and S. Sioutas. “Automatic melodic harmonization: An overview, challenges and future directions”. In: *Trends in Music Information Seeking, Behavior, and Retrieval for Creativity*. Ed. by P. Kostagiolas, K. Martzoukou, and C. Lavranos. IGI Global, 2016, pp. 146–165.
- [Pac] “Markov constraints: steerable generation of Markov sequences”. In: *Constraints* 16.2 (Apr. 2011), pp. 148–172. ISSN: 1572-9354. DOI: [10.1007/s10601-010-9101-4](https://doi.org/10.1007/s10601-010-9101-4).
- [Mie99] Kaisa Miettinen. *Nonlinear multiobjective optimization*. Vol. 12. Springer Science & Business Media, 1999.
- [Mo22] Ying Mo. “Designing an automatic piano accompaniment system using artificial intelligence and sound pattern database”. In: *Mobile Information Systems* 2022 (2022).
- [Niu23] Haina Niu. “Accompaniment Generation Based on Deep Learning and Genetic Algorithm”. In: *IEEE International Conference on Control*,

- Electronics and Computer Technology (ICCECT)*. 2023, pp. 58–65. DOI: **10.1109/ICCECT57938.2023.10141257**.
- [OD01] Nicola Orio and François Déchelle. “Score Following Using Spectral Analysis and Hidden Markov Models”. In: *International Computer Music Conference, ICMC*. Michigan Publishing, 2001.
- [PEB06] Jean-François Païement, Douglas Eck, and Samy Bengio. “Probabilistic Melodic Harmonization”. In: *Advances in Artificial Intelligence: 19th Conference of the Canadian Society for Computational Studies of Intelligence, Canadian AI 2006, Québec City, Québec, Canada, June 7-9, 2006. Proceedings 19*. Vol. 4013. Lecture Notes in Computer Science. Springer, 2006, pp. 218–229. DOI: **10.1007/11766247_19**.
- [PB05] Bryan Pardo and William P. Birmingham. “Modeling Form for On-line Following of Musical Performances”. In: *The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference*. AAAI Press / The MIT Press, 2005, pp. 1018–1023.
- [PATW99] Somnuk Phon-Amnuaisuk, Andrew Tuson, and Geraint Wiggins. “Evolving Musical Harmonisation”. In: *Artificial Neural Nets and Genetic Algorithms*. Vienna: Springer Vienna, 1999, pp. 229–234. ISBN: 978-3-7091-6384-9.
- [PAW+99] Somnuk Phon-Amnuaisuk, Geraint Wiggins, et al. “The four-part harmonisation problem: a comparison between genetic algorithms and a rule-based system”. In: *Proceedings of the AISB’99 Symposium on Musical Creativity*. AISB London. 1999, pp. 28–34.
- [RFV13] Stanisław A Raczyński, Satoru Fukayama, and Emmanuel Vincent. “Melody harmonization with interpolated probabilistic models”. In: *Journal of New Music Research* 42.3 (2013), pp. 223–235.
- [Ran99] Don Michael Randel. *The Harvard concise dictionary of music and musicians*. Harvard University Press, 1999, pp. 10–106.

- [Rap04] Christopher Raphael. “A Hybrid Graphical Model for Aligning Polyphonic Audio with Musical Scores”. In: *International Society for Music Information Retrieval Conference (ISMIR)*. 2004.
- [Rhy+22] Seungyeon Rhyu et al. “Translating Melody to Chord: Structured and Flexible Harmonization of Melody With Transformer”. In: *IEEE Access* 10 (2022), pp. 28261–28273. DOI: **10.1109/ACCESS.2022.3155467**.
- [Rob+18] Adam Roberts et al. “A Hierarchical Latent Vector Model for Learning Long-Term Structure in Music”. In: *International Conference on Machine Learning (ICML)*. Vol. 80. Proceedings of Machine Learning Research. PMLR, 2018, pp. 4361–4370.
- [Sab+21] R. Sabitha et al. “Artificial Intelligence Based Music Composition System-Multi Algorithmic Music Arranger(MAGMA)”. In: *International Conference on Electronics and Sustainable Communication Systems (ICESC)*. 2021, pp. 1808–1813. DOI: **10.1109/ICESC51422.2021.9532706**.
- [SOS04] Diemo Schwarz, Nicola Orio, and Norbert Schnell. “Robust Polyphonic Midi Score Following with Hidden Markov Models”. In: *International Computer Music Conference, ICMC*. Michigan Publishing, 2004.
- [SW20] Ningning Shi and Yingfeng Wang. “Symmetry in computer-aided music composition system with social network analysis and artificial neural network methods”. In: *Journal of Ambient Intelligence and Humanized Computing* (2020), pp. 1–16.
- [SGA20] Maksym Shopynskyi, Nataliia Golian, and Iryna Afanasieva. “Long Short-Term Memory Model Appliance for Generating Music Compositions”. In: *IEEE International Conference on Problems of Infocommunications. Science and Technology*. 2020, pp. 239–242. DOI: **10.1109/PICST51311.2020.9468088**.
- [SMB08] Ian Simon, Dan Morris, and Sumit Basu. “MySong: Automatic Accompaniment Generation for Vocal Melodies”. In: *SIGCHI Conference on Human Factors in Computing Systems*. CHI ’08. Florence, Italy: Associa-

- tion for Computing Machinery, 2008, pp. 725–734. ISBN: 9781605580111. DOI: **10.1145/1357054.1357169**.
- [SDS10] Jon Sneyers and Danny De Schreye. “AOPCALEAPS: Automatic music generation with CHRiSM”. In: Oct. 2010.
- [Spa99] Randall Richard Spangler. “Rule-based analysis and generation of music”. PhD thesis. California Institute of Technology, 1999. DOI: **10.1109/TASLP.2016.2603006**.
- [Sun+21] C.-E. Sun et al. “Melody harmonization using orderless NADE, chord balancing, and blocked Gibbs sampling”. In: *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*. 2021, pp. 4145–4149.
- [Sut+21] Phisetphong Suthaphan et al. “Music Generator for Elderly Using Deep Learning”. In: *Joint International Conference on Digital Arts, Media and Technology with ECTI Northern Section Conference on Electrical, Electronics, Computer and Telecommunication Engineering*. 2021, pp. 289–292. DOI: **10.1109/ECTIDAMTNCN51128.2021.9425691**.
- [TT21] Senem Tanberk and Dilek Bilgin Tükel. “Style-Specific Turkish Pop Music Composition with CNN and LSTM Network”. In: *IEEE World Symposium on Applied Machine Intelligence and Informatics (SAMI)*. 2021, pp. 000181–000185. DOI: **10.1109/SAMI50585.2021.9378654**.
- [Tem11] David Temperley. “The Cadential IV in Rock”. In: *Music Theory Online* 17 (Apr. 2011). DOI: **10.30535/mto.17.1.8**.
- [TDSG18] Lluís L. Trulla, Nicola Di Stefano, and Alessandro Giuliani. “Computational Approach to Musical Consonance and Dissonance”. In: *Frontiers in Psychology* 9 (2018). ISSN: 1664-1078. DOI: **10.3389/fpsyg.2018.00381**.
- [TA99] C.P. Tsang and M. Aitken. “Harmonizing music as a discipline of constraint logic programming”. In: *International Computer Music Conference, ICMC*. 1999.

- [Tsu+17] Hiroaki Tsushima et al. “Function- and Rhythm-Aware Melody Harmonization Based on Tree-Structured Parsing and Split-Merge Sampling of Chord Sequences”. In: *International Society for Music Information Retrieval Conference (ISMIR)*. 2017, pp. 502–508.
- [Wan+20] Ziyu Wang et al. “Learning Interpretable Representation for Controllable Polyphonic Music Generation”. In: *International Society for Music Information Retrieval Conference (ISMIR)*. 2020, pp. 662–669.
- [WC16a] Raymond P. Whorley and Darrell Conklin. “Music Generation from Statistical Models of Harmony”. In: *Journal of New Music Research* 45.2 (2016), pp. 160–183. DOI: **10.1080/09298215.2016.1173708**.
- [Wig+99] G. A. Wiggins et al. “Evolutionary methods for musical composition”. In: *Int. Journal of Computing Anticipatory Systems* (1999).
- [Wil+17] Duncan Williams et al. “Affective calibration of musical feature sets in an emotionally intelligent music composition system”. In: *ACM Transactions on Applied Perception (TAP)* 14.3 (2017), pp. 1–13.
- [WC16b] Yi-Chan Wu and Homer H. Chen. “Emotion-flow guided music accompaniment generation”. In: *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*. IEEE, 2016, pp. 574–578. DOI: **10.1109/ICASSP.2016.7471740**.
- [Yeh+21] Y.-C. Yeh et al. “Automatic melody harmonization with triad chords: A comparative study”. In: *Journal of New Music Research* 50.1 (2021), pp. 37–51.
- [Yi+22] Li Yi et al. “AccoMontage2: A Complete Harmonization and Accompaniment Arrangement System”. In: *International Society for Music Information Retrieval Conference (ISMIR)*. 2022, pp. 248–255.
- [YG07] Liangrong Yi and Judy Goldsmith. “Automatic Generation of Four-part Harmony.” In: *BMA* 268 (2007).
- [YT10] A. Yilmaz and Z. Telatar. “Note-against-note two-voice counterpoint by means of fuzzy logic”. In: *Knowledge-Based Systems* 23.3 (2010), pp. 256–266.

- [Yin+21] Zongyu Yin et al. ““A Good Algorithm Does Not Steal–It Imitates”: The Originality Report as a Means of Measuring When a Music Generation Algorithm Copies Too Much”. In: *Artificial Intelligence in Music, Sound, Art and Design: 10th International Conference, EvoMUSART 2021, Held as Part of EvoStar 2021, Virtual Event, April 7–9, 2021, Proceedings 10*. Springer. 2021, pp. 360–375.
- [Yin+23] Zongyu Yin et al. “Deep learning’s shallow gains: a comparative evaluation of algorithms for automatic music generation”. In: *Machine Learning* 112.5 (2023), pp. 1785–1822.
- [ZL21] Te Zeng and Francis C. M. Lau. “Automatic Melody Harmonization via Reinforcement Learning by Exploring Structured Representations for Melody Sequences”. In: *Electronics* 10.20 (2021). ISSN: 2079-9292.
- [ZX21] Jingwei Zhao and Gus Xia. “AccoMontage: Accompaniment Arrangement via Phrase Selection and Style Transfer”. In: *International Society for Music Information Retrieval Conference (ISMIR)*. 2021, pp. 833–840.
- [ZXW22] Jingwei Zhao, Gus Xia, and Ye Wang. “Domain Adversarial Training on Conditional Variational Auto-Encoder for Controllable Music Generation”. In: *International Society for Music Information Retrieval Conference (ISMIR)*. 2022, pp. 925–932.
- [Zhu+18] Hongyuan Zhu et al. “XiaoIce Band: A Melody and Arrangement Generation Framework for Pop Music”. In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD*. ACM, 2018, pp. 2837–2846. DOI: [10.1145/3219819.3220105](https://doi.org/10.1145/3219819.3220105).
- [Zhu+20] Hongyuan Zhu et al. “Pop music generation: From melody to multi-style arrangement”. In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 14.5 (2020), pp. 1–31.