Politecnico di Torino

POLITECNICO DI TORINO

Master's degree in Management Engineering

Academic Year 2022/2023

Melody harmonization through Integer programming

Supervisor

Prof. Fabio SALASSA

Prof. Elena RENER

Candidate

Marco ZOLLO

April 2024

Abstract

This work presents a model that utilizes exact combinatorial optimization methods to generate the harmonization of a given four-four melody in the key of C major/A from a proposed set of chords. A linear programming formulation has been developed with the objective of identifying the most favorable sequence of chords from a given set. This is achieved by considering a matrix of transition costs based on rules of harmony theory and commonly used chord progressions in Pop music. The resulting musical accompaniment is integrated with the melody and visualized using MuseScore4.

To assess the model's outputs, a comparison has been conducted between the chords used in well-known songs and the chords chosen by the model for those songs. If developed, the proposed model could serve as an educational tool for composers and provide support in discovering new ideas for accompaniment to complement their melodies.

ACKNOWLEDGMENTS

Table of Contents

Chapter 1

Introduction

1.1 Motivation

The generation of music through artificial intelligence tools and composition using algorithms has garnered significant interest in recent times. The availability of digital computing has greatly expedited the exploration of algorithmic composition systems. This increase in research is driven by a combination of curiosity and efforts to analyze and this field of research.

This work is driven by the recognition that automatically generated guitar solos can serve as a valuable resource for composers facing the challenge of generating a chord progression into their musical pieces, without an in-depth understanding of the instrument.

The main intent of this work is to delve into the possibility that mixed-integer programming can emerge as a valid tool for addressing the complexity of melodic harmonization. The primary objective is to gain a comprehensive understanding of the effectiveness of this methodology, clearly highlighting its inherent limitations and advantages in comparison to other well-established musical techniques. The decision to focus on mixed-integer programming stems from the ambition to explore new perspectives within the realm of algorithmic composition, specifically honing in on its adaptability to the context of melodic harmonization. In this context, the analysis of the limitations and advantages of mixed-integer programming emerges as a fundamental step towards a comprehensive examination of this melodic harmonization methodology.

For composers unfamiliar with harmony theory, this technology offers a creative solution, using the potential of automated generation to decide a chord progression for the melody given by users. Furthermore, our motivation extends to the educational realm, where automatically generated chords becomes a potent pedagogical aid. This work envisions assisting amateur musicians in understanding musical

theory in a more practical way.

1.2 Problem

This document addresses the harmonization of melodies, a key challenge in the field of music generation. The primary goal is to generate a coherent sequence of chords that complements a given sequence of input notes through the resolution of an optimization problem. In formulating the model and subsequently solving it using Python programming, various issues were encountered, and some of them are discussed below. It is noteworthy that the problem was modeled by drawing parallels to a minimum path problem.

- Rhythmic Limits: Setting rhythmic limits presents a delicate challenge. Deciding on the time signature for the model is the initial step, with the difficulty lying in the fact that using only 4/4 time significantly restricts rhythmic harmonization possibilities. Additionally, determining the number of chords per measure and their durations adds complexity.

- Choice of Key or Mode: Choosing the musical key or mode for the model poses another challenge. In our approach, we introduced specific constraints, confining the harmonic context to the C major scale and its relative minors. This decision simplifies the problem's complexity, directing attention to a specific harmonic context with a clear and universally recognized chord structure.

- Weight Assignment for Edges: Determining the weights assigned to edges connecting chords in the graph, representing harmonic progressions, required significant effort. These choices, though based on musical conventions, lack objectively valid rules. Imbalanced weightings could hinder the model's ability to find interesting harmonic solutions.

- Handling XML Music Files: Implementing the mathematical model into a functional application posed significant challenges. Managing musical XML files, specifically extracting melody from these files, and subsequently writing the generated chords onto them for user display required intricate handling of libraries.

- Testing Phase: One of the critical operations was the testing phase. This is because, given that the theory of harmony is not guided by objectively valid rules, evaluating the compositions generated by the model becomes highly complex and subjective. For a given composition, it could be positively assessed by one person and negatively by another, depending on individual taste.

In summary, while formulating the mathematical model posed challenges in constraint selection, rhythmic considerations, and harmonic conventions, implementing the model as a functional application introduced additional complexities in handling musical XML files and how to write the costrains defined in the mathematical model. Addressing these challenges required a nuanced and comprehensive approach to ensure effective melody harmonization.

1.3 Methodology

Our strategy involves conceptualizing the melody harmonization as an optimization problem. Chords are represented by nodes within a graph, framing the harmonization task as the one of finding a minimum path in a weighted graph.

The idea of structuring the problem in this way, rather than using machine learning algorithms, is inspired by a few earlier papers that tackled music generation through combinatorial optimization, specifically referencing works such as "Describing Global Musical Structures by Integer Programming on Musical Patterns" and "Generating guitar solos by Integer Programming" [1]. In the latter, guitar solos are generated from a set of solo segments, known as licks. The model, in this case, adds selected licks through the search for a minimum path, generating a melody for a predefined chord progression, particularly a 12-bar blues in C major. In our work, we essentially address the opposite problem: choosing chords to accompany a melody that is not fixed, as it is input by the user. The harmonic context in which the model is designed is limited to the C major scale, its relative natural, harmonic, and melodic minors. The model's functionality can be easily applied to all keys by transposing the chords. Let's look more specifically at the methods used to approach the model formulation, translation into programming language and testing phases. The optimization problem is formulated as a minimum path problem, where nodes represent chosen chords at various time instances, and edges represent transition costs between chords at specific time points. Transition costs are stored in matrices, where the element (i, j) represents the cost from chord i to j. The objective is to minimize the cost function by selecting the most favorable chord progression. Preprocessing involves choosing individual transition costs based on tonal music theory rules. About the coding, the model translation into Python was accomplished using the MIP library, allowing us to encode and solve the optimization problem efficiently. We transformed the mathematical model into executable code, utilizing specialized solvers for optimal outcomes. Simultaneously, we read the user-input melody from a file in Python and incorporated chosen chords using the Music-21 library [2]. This process involved seamlessly modifying the existing melody to align with both theoretical principles and user preferences. The

integration of tonal music theory rules ensured the model's compatibility with musical conventions, facilitating the generation of harmonizations in a straightforward and effective manner.

Then, the testing phase involved a systematic and iterative approach, focusing on well-known musical pieces to evaluate the model's effectiveness in real-world scenarios. Melodies were extracted from selected pieces, serving as input for the model to generate harmonizations. Comparative analyses were then conducted, where the model-generated harmonizations were compared with the original accompaniments of the chosen test pieces. Specifically, tests were conducted for each piece of music, varying the model parameters to identify the optimal solutions and understand their impact on the model's performance.

1.4 State of art

In this chapter, our aim is to provide a comprehensive overview of the multifaceted challenges inherent in the field of music generation. We will delve into the intricate components of the music generation problem, encompassing the creation of harmonies, melodies, and rhythmic structures. Furthermore, we will explore the diverse technologies employed in the realm of music generation, shedding light on the tools and methodologies that play pivotal roles in the creative process.

Our discussion will extend to an examination of the most significant studies and software applications within this domain.

Through this exploration, we aim to offer a nuanced understanding of the state of the art, encompassing the varied dimensions of music generation and the influential studies and software shaping its trajectory.

The field of music generation has witnessed significant advancements in recent years, driven by a diverse array of methodologies aimed at unraveling the complexities of automatic composition.

The scope of automatic music generation is very broad and there are many aspects of it that can be addressed; the main challenges faced in this area are the following:

- Melody generation constitutes one of the earliest aspects of music subject to automatic creation. When considering the problem of music generation, the simplest form of this exercise that comes to mind is the composition of monophonic melodies without accompaniment. Anyway, in most melody generation systems, the goal is to compose melodies with characteristics similar to a chosen style, such as Western tonal music or free jazz or melodies for a certain chord progression.
- Harmony generation refers to the simultaneous sounding of different pitches to create chords and chord progressions. Generating harmonic progressions

automatically is another popular aspect of music generation, particularly when aiming to complement a given melody or fit within a specific musical genre.

- Rhythm generation involves the creation of rhythmic patterns without direct human intervention. This process is crucial in music composition, as rhythm forms the backbone of musical structure and provides the framework for melodic and harmonic elements to interact.

In the field of music generation, beyond addressing specific challenges in harmony, melody, and rhythm generation, there may be additional distinct problems to tackle. Alternatively, in some cases, efforts are made to address the combination of various subproblems.

In addition to the various sub-problems, there is an additional distinction to highlight. In the realm of automatic music generation, the described sub-problems or combinations thereof are addressed through different methodologies outlined below.

1.4.1 Machine Learning

In the context of music generation involves extracting new musical elements from existing musical data. Advanced models incorporate sophisticated musical features such as pitch contour and rhythmic patterns. In addition to conventional tasks like prediction, classification, and translation, deep learning is increasingly recognized as a viable option for music generation. This is evident in projects leveraging deep learning's capabilities to automatically learn musical styles from diverse musical pieces, enabling the generation of samples. However, the direct application of deep learning to music generation presents challenges related to control, structure, creativity, and interactivity.

This technique relies on artificial neural networks with multiple layers processing complex abstraction levels automatically extracted from the data. The evolution has led to the widespread use of networks like LSTM and Generative Adversarial Networks (GANs) to address specific challenges in music generation.

While these architectures are powerful, they often lack direct control over the generation process and operate autonomously without human interaction. The direct implementation of deep learning can result in content that mimics the training dataset without showcasing true creativity. This poses challenges in understanding the exact functioning of the generation process and controlling the system's output.

1.4.2 Genetic Algorithms

Genetic algorithms, also known as evolutionary algorithms, simulate the natural evolutionary process by starting with an initial population of random solutions to an

optimization problem. The individuals in the initial population, which in our case are random solutions to the specific problem of music generation we are addressing, undergo evaluation. Often, assessing musical quality, or "fitness," poses an intrinsic challenge, as precisely defining what constitutes an optimal composition can be complex. The next step is the so-called Crossover: here, the selected "individuals" following the evaluation are indeed "crossed." This step simulates natural genetic crossover. In some individuals of the new generation, random mutations occur. This process introduces random variations in the solutions, contributing to genetic diversity. The new generation of individuals replaces the previous population, and the preceding steps of "Evaluation" and Crossover are repeated for a certain number of iterations (fixed or variable) until an acceptable final solution is achieved.

1.4.3 Rule Based

An algorithm of this type follows a series of predetermined musical rules to create compositions. This approach is based on formalizing the rules of music theory that guide the compositional process. Hiller and Isaacson (1957) are pioneers in this approach, and their systems adhere to strict musical rules for generating compositions, as seen in the model by Schottstaedt (1984) for counterpoint generation.

The rule-based algorithm in music generation can be implemented in various ways, often combining rules with optimization techniques to ensure adherence to all rules during the creative process. These systems may start from initial material, such as existing musical sequences or random inputs, which are then refined through the application of musical rules.

Rules can be implemented using Constraint Programming, a declarative approach well-suited for describing music theory rules. The complexity in designing such systems lies in the need to encode a significant number of rules, balancing formal definition and stylistic variety. Additionally, there is a challenge in finding the right balance between adding rules to better fit the modeled style and limiting restrictions to be more open to various musical styles.

This approach can lead to more efficient exploratory creativity, although it may limit the variety of the output due to the imposition of specific constraints.

1.4.4 Markov Models

It is a stochastic model describing a sequence of possible events in which the probability of each event depends only on the state attained in the previous event.

In the musical context, Markov chains serve as a tool for modeling the sequential transition between musical notes or events. These stochastic processes, characterized by a finite number of states, define the transition between states through a transition

matrix. The probabilities of transitioning from one state to another are based on the relative frequencies in the musical corpus used for training.

The introduction of an n-order approach allows the influence of the last n states on the probability of the next transition, adding greater complexity to the modeling of musical sequences.

Despite their fundamental effectiveness in the sequential representation of music, the use of Markov chains can sometimes result in repetitive and less creative outcomes, especially with higher orders. Nevertheless, the integration of Markov chains with other techniques at higher structural levels, as seen in the generation of entire compositions, can contribute to more intricate and creative results.

1.4.5 Optimization Problem

The optimization approach in music generation treats composition as a combinatorial optimization problem, providing significant advantages in setting constraints on long-term musical structure and cadences. This methodology combines optimization techniques with in-depth knowledge of music theory to impose a broader structure, such as the 12-bar blues, and specific constraints on musical cadences. This approach effectively models music generation by incorporating complex musical rules, contributing to defining a more coherent and musically satisfying outcome. The fusion of optimization and theoretical knowledge opens up new creative possibilities, allowing exploration of a wide range of musical constraints and guiding the generation process toward musically meaningful results.

Each approach brings a unique set of tools and insights to the forefront, contributing to a rich tapestry of methods designed to address the multifaceted challenges inherent in music generation. These techniques are not mutually exclusive and can be combined in various ways to create innovative music generation systems that contribute to the evolving landscape of musical creativity.

1.5 Tools for Music Generation

In the evolution of music, machine learning, optimization techniques, genetic algorithms, and rule-based approaches have played a significant role in the development of tools capable of generating solutions to the sub-problems of musical generation:

- Melody generation;
- Rhythm generation
- Harmony generation.

These tools, or models, employ a variety of techniques and technologies to create unique musical compositions. Below, we will explore some of these tools, examining how they work and how they have influenced the field of music generation.

The following list of tools has been compiled with gratitude to the following papers that provide an in-depth analysis of the state-of-the-art in music generation:

- Functional Taxonomy of Music Generation Systems [3]
- Computational Creativity and Music Generation Systems: An Introduction to the State of the Art [4]
- A Systematic Review of Artificial Intelligence-based Music Generation: Scope, Applications, and Future Trends [5]
- A Survey of AI Music Generation Tools and Models [6]
- The Pinkerton model, also called the "Banal Tune Maker," was developed by Pinkerton in 1956 [7] to generate melodies inspired by existing nursery rhymes. It utilized Markov models, analyzing transitions between notes in a corpus of 39 children's melodies to create a transition matrix. This matrix represented the probabilities of transitioning between different musical states, such as notes in the diatonic major scale of C and symbols for pauses or sustained notes.

  During melody generation, the model followed a random walk process, selecting the next note based on the probabilities defined in the transition matrix. However, the Pinkerton model's simplistic approach, where each note's probability depends solely on the preceding note, resulted in melodies lacking complexity and originality. Despite its early use of Markov models, the Pinkerton model's limitations highlight the need for more sophisticated techniques to generate musically interesting melodies.

- GenJam is an interactive genetic algorithm designed for learning jazz improvisation [8]. It employs two hierarchically correlated populations to represent melodic ideas at both the measure and phrase levels. These populations evolve through tournament selection, single-point crossover, musically significant mutation, and replacement with a 50% generational gap. The fitness of individual measures and phrases derives from real-time feedback provided by a human mentor while GenJam improvises, accompanied by a synthesized rhythm section.

  GenJam has been utilized in live performances under the name Al Biles Virtual Quintet, featuring the author on trumpet and GenJam playing a variety of synthesized instruments. Together, they perform a repertoire spanning over 90

compositions in various jazz, Latin, and new age styles. Recent enhancements include a pitch-to-MIDI capability, enabling GenJam to listen to a human soloist, map their four-bar phrases to GenJam's genetic representation, apply selected mutation operators to these phrases, and play them in real-time while exchanging quarters or eighths with a human soloist. This makes GenJam genuinely interactive both during training and performances.

- The Marques Music Generator aims to autonomously produce high-quality musical compositions without external user intervention, starting sequences, or subjective feedback [9]. It utilizes algorithmic composition methods derived from the study of rules followed by human composers, incorporating evolutionary algorithms, specifically a variant known as "Familial Competition." The algorithm begins with a random population in the search space, applying crossovers and mutations to create new individuals (children) that are evaluated against their parents. The two best individuals proceed to the next generation. This process continues until a predefined fitness value is achieved or a set number of iterations is reached, enhancing the effectiveness compared to standard genetic algorithms.

- The "Melisma Stochastic Melody Generator" is an advanced tool for generating melodies using artificial intelligence [6]. It incorporates randomness into the melody creation process through a stochastic process, allowing for the generation of unique tunes each time. Users can customize the specifications for the tune generation at various levels, providing a high degree of personalization. The generator uses a fitness function to evaluate the quality of the melodies it generates. This function is based on musical rules and principles, ensuring that the generated melodies are musically coherent. An evolutionary algorithm, which mimics the process of natural evolution, is used to evolve and optimize the melodies over time. The generator continues to iterate and improve upon the melodies until it achieves a satisfactory level of fitness or until a certain number of iterations have been completed. In essence, the Melisma Stochastic Melody Generator combines artificial intelligence and music theory to create unique and pleasing melodies from scratch.

- MorpheuS is an automatic music generation system that addresses the challenge of creating music with long-term structure. It uses advanced pattern detection techniques to identify repeated sequences in a given example piece of music (the template). These patterns represent recurring musical structures. The identified patterns are then used to constrain the process of generating a new polyphonic composition. In other words, MorpheuS relies on existing patterns to create new coherent musical parts [10]. The music generation

process is guided by an efficient optimization algorithm called "variable neighborhood search", which uses a mathematical tension model as its objective function. Tonal tension is a musical concept related to harmonic stability and instability. MorpheuS also has the capability to generate music based on a specific tension profile, allowing it to create musical parts that fit different emotions or situations, such as in game or film music contexts. Compositions generated by MorpheuS have been performed in live concerts. In summary, MorpheuS combines pattern analysis, optimization, and tension models to create structured and engaging music. It's an intriguing tool for composers and those interested in exploring new creative possibilities in music.

- The Continuator is a system developed by François Pachet at the Sony Computer Science Laboratory in Paris, which operates based on Markov models[11] . This system represents a significant advancement in the field, as it successfully bridges the gap between two traditionally incompatible types of musical systems: interactive musical systems and music imitation systems.

  Interactive musical systems, while engaging for the user, have historically been limited in their ability to generate stylistically consistent material. On the other hand, music imitation systems, though capable of producing stylistically consistent material, are fundamentally not interactive. However, the Continuator manages to combine the strengths of both types of systems.

  The Continuator enhances the technical abilities of musicians by providing them with stylistically consistent material that the system learns automatically. This is achieved by constructing operational representations of musical styles in real-time. The system employs a Markov model of musical styles, enhanced to handle musical elements such as rhythm, beat, harmony, and imprecision. Consequently, the Continuator can learn and generate music in any style, both autonomously and as continuations of the musician's input, or as support for interactive improvisation.

  Furthermore, the design of the Continuator opens up new possibilities for collaborative musical play. The system has been tested in various real-world contexts, demonstrating its practical applicability and effectiveness. For example, in a study at the University of Bologna where the use of this system was introduced to children, positive educational effects were observed[12].

  In conclusion, the Continuator represents a significant step forward in the field of music generation, offering a unique combination of interactivity and stylistic consistency that sets it apart from other systems.

- MuseNet, a project by OpenAI, represents a state-of-the-art technology in the field of music generation [13]. It employs a deep neural network to generate musical compositions, capable of creating 4-minute compositions with up

to 10 different instruments, and can blend styles ranging from country to Mozart to the Beatles. Rather than being explicitly programmed with our understanding of music, MuseNet has discovered patterns of harmony, rhythm, and style by learning to predict the next token in hundreds of thousands of MIDI files. It utilizes the same general unsupervised technology as GPT-2, a large-scale transformer model trained to predict the next token in a sequence, whether it's audio or text. However, MuseNet does have limitations. For instance, the instruments you request are strong suggestions, not requirements. MuseNet generates each note by calculating probabilities over all possible notes and instruments, and while it shifts to make your instrument choices more likely, there's always a chance it might choose something else. MuseNet also struggles with unusual pairings of styles and instruments, such as Chopin with bass and drums. Despite these limitations, MuseNet represents a significant advancement in the application of AI for music generation.

- Riffusion is an artificial intelligence model that generates music using spectrograms, visual representations of sound [14]. These spectrograms are generated by an image synthesis model called Stable Diffusion 1.5, which has been adapted to work with sound.

  Once the model generates the spectrogram, Riffusion converts it into an audio file using an inverse Fourier transformation. This process transforms the image of sound into real audio that we can listen to.

  Riffusion can also create new musical compositions by combining different sounds together. This is done using a feature of the Stable Diffusion model called img2img, which allows interpolation between different images (or in this case, sounds).

  Finally, although the audio files generated by Riffusion are usually short, the model can create longer compositions by stringing together these short segments. It is a tool that can be used to explore new forms of musical expression.

- MUSICGEN is an artificial intelligence model designed for high-quality music generation based on textual descriptions, such as "90s rock song with a guitar riff [15]. The model leverages recent advancements in self-supervised audio representation learning, sequential modeling, and audio synthesis. To address the challenges of music generation, MUSICGEN represents audio signals as multiple streams of discrete tokens, allowing for both high-quality audio generation and effective audio modeling. The model produces coherent and high-quality music, considering both the provided textual description and the desired melodic structure.

- CHORAL is an expert system created by Kemal Ebcioglu for harmonizing four-part chorales in the style of J.S. Bach through a Rule-based alghoritm [16]. With over 270 rules expressed in a first-order predicate calculus, CHORAL uses a rule-based approach to make decisions in chorale harmonization. It considers chordal structure, individual melodic lines, and voice leading, employing an intelligent backtracking method. The system is implemented in the BSL programming language, designed specifically for CHORAL. This approach ensures coherent harmonizations consistent with Bach's style.

- Magenta is designed to handle the entire music composition process. Leveraging advanced techniques such as recurrent neural networks and generative adversarial networks, Magenta has the capability to compose original music spanning various genres and styles[17].

  Magenta is not limited to melody generation; it can also create harmonies and rhythms, allowing it to generate complete songs autonomously. Additionally, Magenta can harmonize melodies, generate chord progressions, and create rhythmic patterns suitable for different musical contexts. Its interactive tools enable real-time collaboration between users and the AI model, empowering users to influence the direction of music generation.

  Overall, Magenta represents a significant advancement in the field of music composition through artificial intelligence. By integrating cutting-edge machine learning algorithms with interactive user interfaces, Magenta expands the possibilities of creative expression in music composition.

- The APOPCALEAPS system is a pop music generation system that utilizes probability rules implemented in CHRiSM, a high-level programming language specifically designed for writing constraint solvers [18]. It assigns a chord per measure and models the chord sequence as a simple Markov chain. Transition probabilities are set manually based on personal taste and experience but can also be learned automatically from examples. The input specifies the key and the number of measures, and the chord sequence in the output is encoded as constraints mchord(X, C), where C is the chord for measure X. If a piece is in a major key, the first and last measures get the "C major" chord, whereas if it is in a minor key, they get "A minor" chords.

- The "Piano Impainting" application is based on machine learning technologies, specifically on linear encoder-decoder transformers. This allows PIA to efficiently restore missing regions in a piano performance.

- "MidiNet" is a network designed for music generation in the symbolic domain. It employs neural networks to transform random noise input into a musical output [19]. This process is guided by a generative model that aims to mimic

the training data. This enables the model to generate melodies from scratch, follow a chord sequence, or be conditioned on the melody of preceding bars.

- JamBot is an artificial intelligence tool that creates polyphonic music, meaning music involving multiple notes played simultaneously [20].

  It is based on a technology called LSTM (Long Short-Term Memory), a type of artificial neural network used for machine learning. JamBot operates in two phases: first, it predicts a chord progression based on a chord embedding, which is a numerical representation of chords. Subsequently, it generates polyphonic music based on the predicted chord progression.

- The "Pop Music Transformer" is a machine learning model developed for generating pop music compositions for the piano [21]. During the generation process, the model produces a complete musical composition that includes both the melody and chord structure. Additionally, the model provides controllability over local tempo changes and chord progression, offering a certain degree of flexibility and variety in the generated music.

1.6 Outline

This outline give a structure to the subsequent chapters and their respective topic. The second chapter aims to explain harmonization theory essential for comprehending the project. The third chapter delves into the idea behind the mathematical model and show its structure. Additionally, it outlines the specific components and methodologies used in shaping the model as variables, target function etc. The fourth chapter is dedicated to explaining the translation of the mathematical model into Python and also, how to handle musical files through Python libraries. This section focuses into the practical implementation of the model, emphasizing the coding aspects and the utilization of Python-based tools for effective execution. Moving on to the fifth chapter, the focus shifts to testing the model and presenting the results of harmonizing well-known musical pieces. A comparative analysis with the original compositions is conducted, showcasing the effectiveness and performance of the model. These tests involve systematic variations of the model's parameters, providing a comprehensive evaluation of its adaptability and outcomes.

Chapter 2

Background in Music Theory

2.1 Introduction to Musical Harmony

This chapter delves into the foundational aspects of tonal harmonic theory to provide a theoretical framework essential for comprehending the underlying motivations behind constructing the model. As outlined in the previous section, the central aim of this paper is to find a methodology for harmonizing a melody supplied by the user. In musical terms, a "melody" is defined as a sequential arrangement of individual notes, creating a linear progression. Melodies play a crucial role in shaping the character and emotional impact of a piece of music. They are typically the most recognizable musical element in a piece of music, providing a sense of structure, identity, and emotional expression. The harmonization of a melody refers to the process of combining chords and melody. The combination of these two elements add complexity and unique overall sound to the song. The result of this process is influenced by the following factors: 1. Relationship between harmony and chords; 2. Progression of chords and cadences.

In the exploration of musical harmony and the harmonization of melodies, it is necessary to understand the fundamental elements of chords and scales. In this work, the application works using exclusively C major scale, which is the starting point to harmonization of a melody.

2.2 C Major Scale Construction

In this section, the construction of the C major scale will be discussed. This scale acts as a fundamental framework, guiding our understanding of musical harmony.

Constructed with a specific formula:

- $(Whole - Whole - Half - Whole - Whole - Whole - Half)$

In music theory, a half step (H) represents the distance of one semitone. It is the smallest interval in traditional Western music. For example, moving from C to C# or from E to F on the piano involves a half step. If we consider the piano keyboard, moving from one key to the very next key – white or black – constitutes a half step. A whole step (W) corresponds to the distance of two semitones. So, when we apply the formula to construct the C major scale we are essentially defining the sequence of intervals in terms of semitones [22]. The figure below shows the resulting sequence of notes.

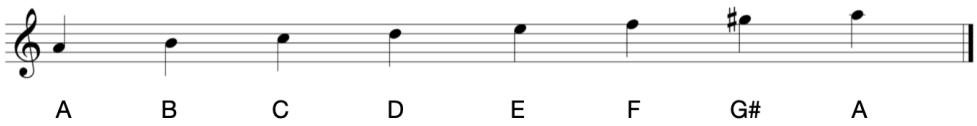

Figure 2.1: C major scale with his formula

The C major scale, with its pure and unambiguous tonal structure, provides a solid starting point for our exploration. By delving into the intervals between each note, we lay the groundwork for understanding how chords can be built and how harmonic progressions can be shaped. This fundamental knowledge of the C major scale will prove essential as we navigate the complexities of harmonizing melodies in the subsequent chapters.

2.2.1 Relative Minor Scale

Minor scales, intricately linked to their major counterparts, add a layer of richness to our exploration of harmonization. The process of deriving minor scales from major scales unveils a fascinating interplay of tones and intervals. Applying the formula:

- $(Whole - Half - Whole - Whole - Half - Whole - Whole)$

to A results in the creation of the A minor scale: A, B, C, D, E, F, G, A [22].

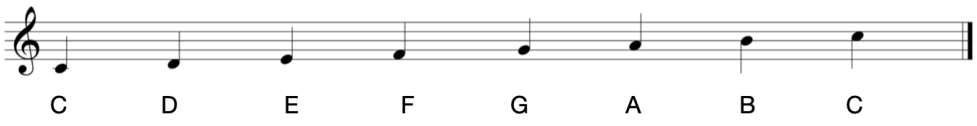

Figure 2.2: A minor scale, relative minor of C major scale

Despite the altered order of notes, a clear connection emerges between the A minor scale and the C major scale. This relationship exemplifies the harmonic duality between major and minor scales. The harmonic richness introduced by minor scales sets the stage for our exploration of chord construction and the subsequent harmonization of user-provided melodies.

The model has been further enriched by the harmonic and melodic A minor scales.

2.2.2 A Harmonic Minor Scale

The A harmonic minor scale introduces an impactful alteration to the natural minor scale. This modification occurs by raising the seventh degree of the scale by a semitone. In the case of the A harmonic minor scale, the G is elevated to G#. This adjustment creates a distinctive and somewhat exotic tonal flavor, enhancing the model's capacity for harmonic diversity. The harmonic minor scale is particularly valuable in creating tension and building anticipation, adding layers of complexity to chord progressions[23].

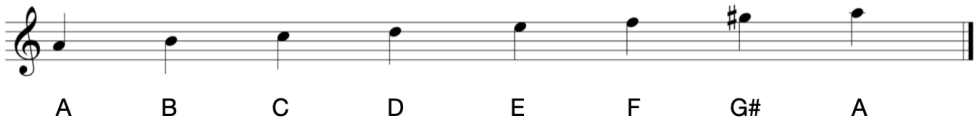

Figure 2.3: A Harmonic minor notes

2.2.3 A Melodic Minor Scale

Similarly, the A melodic minor scale contributes another dimension to our harmonic exploration. This scale alters the sixth and seventh degrees compared to the natural minor scale. Both the F and G are raised by a semitone when ascending, reverting to their natural positions when descending [23].

By including both the melodic variations of the A scale our model expands the range of tonal options it can explore.

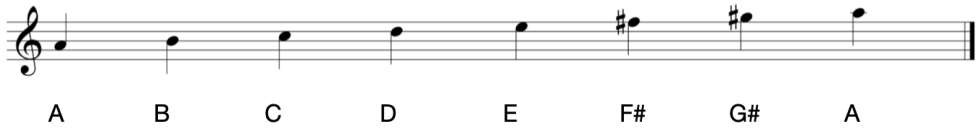

Figure 2.4: A Melodic Minor Scale

2.3 Constructing Chords from Scales

The process explained in this paragraph is the one of constructing chords from the C major scales. Chords are defined as simultaneous combinations of three or more notes that are played together to create a sound. In music theory, chords serve as the basic building blocks of harmony and are crucial for creating the overall tonal structure of a musical piece. Since this work has the aim to add a chord progression to a given melody, it is crucial to understand how they are constructed and how chords are related to single notes.

2.3.1 Triads

The simplest class of chords employed by the model are the “triads”; as suggested by the name are constructed using three notes played simultaneously. The process of constructing triad involves superimposing notes with specific interval distances. To form triads, a precise process based on the structure of the reference scale, in our case, the C major scale, is followed. For each chord, a starting note is selected, often called the "root" of the chord, and from this note, two third intervals are added. The first third interval determines whether the chord will be major or minor. If the third interval consists of two whole tones, a major chord is obtained. If the interval is composed of a tone and a half, a minor chord is obtained. The second third interval, added to the note resulting from the application of the first interval, determines the fifth of the chord. The fifth can be "perfect" (an interval of three and a half tones from the root), "augmented" (an interval of four tones), or "diminished" (an interval of three tones).

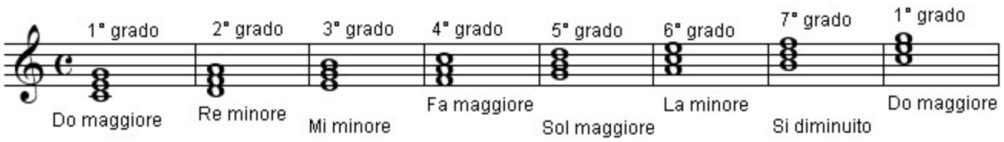

Figure 2.5: A harmonic minor chords

Major Triad

The major triad, with its root, major third, and perfect fifth, emanates a bright and uplifting sound. Known for its sense of stability and consonance, the major triad is often associated with positive and joyful musical expressions. Its harmonically

rich quality contributes to a feeling of resolution and completeness, making it a foundational element in various musical genres.

Minor Triad

In contrast, the minor triad, formed by the root, minor third, and perfect fifth, carries a more subdued and introspective character. The minor third introduces an element of tension, evoking emotions ranging from melancholy to contemplation. Minor triads are frequently employed to convey complex and nuanced feelings, adding depth and emotional diversity to musical compositions [22].

Sus4 Triad

The "Sus4" triad, featuring the root, perfect fourth, and perfect fifth, introduces a sense of suspension and ambiguity to the harmonic landscape. The absence of the third creates an open, unresolved quality, often described as having a "suspended" or "floating" sound. "Sus4" chords are versatile, providing a departure from traditional major or minor tonalities and offering a platform for creative and experimental musical exploration.

Triads exhibiting a minor third and a diminished fifth fall into the category of "diminished" chords. It is characterized by a unique and distinct sound that can be described as tense, unstable, and dissonant. In major scales, a consistent pattern emerges in the formation of chords based on each degree. Specifically, on the 1st, 4th, and 5th degrees, major chords are invariably constructed. On the 2nd, 3rd, and 6th degrees, minor chords are consistently formed. Notably, the 7th degree gives rise to a diminished chord. Taking the specific example of C major, the chords are: C major, F major, G major, D minor, E minor, A minor, and B diminished. The same chords can be obtained doing the same process with the A minor scale [22].

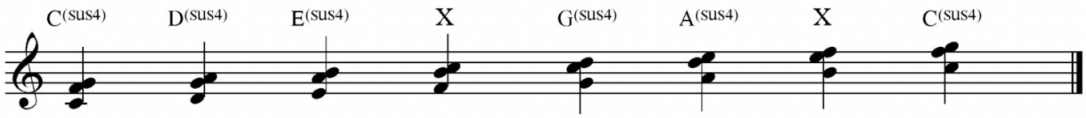

Figure 2.6: Sus4 triad in C major

2.3.2 Seventh Chords

Similar to triads, the "seventh chords" constitute a more complex class of chords utilized by the model. To construct seventh chords, the same foundational process

is applied, but with the incorporation of an additional interval. For each chord, a starting note, referred to as the "root," is chosen, and from this note, two third intervals are added as well as a seventh interval [22] .

The process of adding a seventh interval to triads results in the formation of three distinct types of seventh chords:

- The major seventh chord which are characterized by the root, major third, perfect fifth, and major seventh;
- The minor seventh chord, formed by the root, minor third, perfect fifth, and minor seventh;
- The dominant seventh chord, with its root, major third, perfect fifth, and minor seventh. Its distinct sound often leads to resolutions
- The diminished seventh chord, featuring the root, minor third, diminished fifth, and diminished seventh. It is characterized by dissonant and unstable quality.

Their qualities contribute to the overall richness and diversity of harmonic expressions in our model. In a major scale, specific degrees give rise to different types of seventh chords, adding depth and sophistication to the harmonic palette. On the 1st and 4th degrees, major seventh chords are formed, characterized by a major seventh interval. The 5th degree yields the dominant seventh chord, recognized for its pivotal role in creating tension and leading to resolutions. Meanwhile, the 2nd, 3rd, and 6th degrees consistently generate minor seventh chords, contributing a milder and introspective quality. The 7th degree results in the formation of half-diminished seventh chords, known for their distinctive semi-diminished sound.

In the context of the C major scale, this translates to the creation of specific seventh chords:

- The 1st degree (C) forms a major seventh chord (Cmaj7).
- The 4th degree (F) also produces a major seventh chord (Fmaj7).
- The 5th degree (G) gives rise to a dominant seventh chord (G7).
- The 2nd degree (D), 3rd degree (E), and 6th degree (A) contribute minor seventh chords (Dm7, Em7, Am7).
- The 7th degree (B) generates a half-diminished seventh chord (Bm7b5), infusing a touch of tension and complexity into the harmonic landscape.

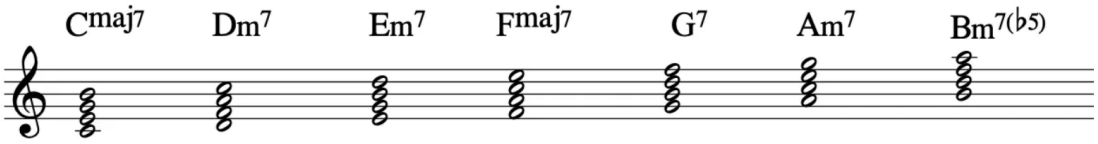

Figure 2.7: Seventh chords in C major

2.3.3 Inversion Chords

The inversion of a chord with the seventh, fifth, and third at the bass is a harmonic technique that involves rearranging the order of notes within the chord In these chords, the lowest note is not the root, creating a diverse harmonic effect. For example, let's consider a C major chord in its root position, consisting of the notes C (root), E (third), and G (fifth). In the inversion with the third at the bass, the order becomes E (root), G (fifth), and C (third). In the inversion with the fifth at the bass, the order becomes G (root), C (third), and E (fifth). Now, considering a C major seventh chord (Cmaj7) in its root position, composed of the notes C (root), E (third), G (fifth), and B (seventh). In the inversion with the seventh at the bass, the order becomes B (root), C (seventh), E (third), and G (fifth)[23].

2.3.4 Dominant Seventh Chords and Substitutions

The last typology of chords that have been used for this work are Dominant Seventh one. In music theory, the use go this chords is often associated with the concept of "temporary modulation," meaning a momentary change of key in the piece of music. In this case, the additional chords are C7, D7, E7. The reason for these substitutions is explained below:

The inclusion of D7 in a C major progression is justified through the concept of a "secondary dominant." In this context, the G chord is reinterpreted not as the V degree of the C major scale but rather as the I degree of a G major scale. The fifth degree (V) of the G major scale aligns precisely with the seventh dominant chord, D7. Doing the same process with F major chord, C major dominant seventh is found.

The use of the E7 chord is justified by the use of the harmonic minor scale within our model. This variation from the natural minor scale ensures that, just like in the major scale, a dominant seventh chord is formed on the V degree, namely E7. The harmonic tension created by these types of chords generally prepares the ear for resolution to their respective I degree. Next paragraphs will discuss how chords played consecutively create tension and resolution. The figure below show A minor harmonic chords, E7 included[24].

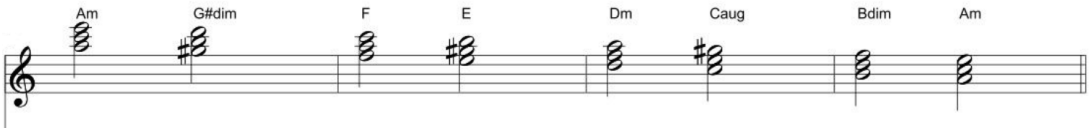

Figure 2.8: C major chords

2.4 Cadences

An important aspect to keep in mind is the relationship that chords have with each other when played in sequence. This is what is called a cadence, and it has an impact on the tension of a piece of music. This aspect has a significant impact on the perception and emotion of a musical piece. They influence the harmonic structure and contribute to creating tension and resolution, providing direction and coherence within the composition.

To better understand the software implementation choices, the cadences taken into consideration are explained below:

- The Perfect Authentic Cadence produces a greater sense of conclusion and consists of the succession of the dominant seventh chord (G7), which creates tension, and the I chord (C), which, when preceded by the V chord, gives a strong sense of resolution to the piece. A similar effect is achieved by the succession of the V chord of the harmonic minor scale and the chord constructed on the I grade (E7 —>Am). In fact, in this scale, the VII degree is raised by a semitone, creating a resolution of tension similar to that of the major scale. The same effect is obtained playing the following chord pairs: C7 - F and D7-G using the concept of "Secondary Dominant" explained in the previous chapter.
- The Imperfect Authentic Cadence is the succession of the VII and I chords. These lead to a strong sense of resolution, especially when the note C and B semi-diminished chord sounds together.
- A "Half Cadence" is any chord progression that goes to V (G7): most common ones are I-V, ii-V, and IV-V. It creates a weak point of pause and can be imagined as the comma of spoken language.
- The "Plagal Cadence" in C major scale uses the progression from IV to I, which means F to C. The feeling given is similar to that of an authentic cadence but softer.
- The "Deceptive Cadence," instead, follows the V chord with a chord other than the I chord. The listener would expect the C major chord, but a different one arrives, creating an unexpected sound.

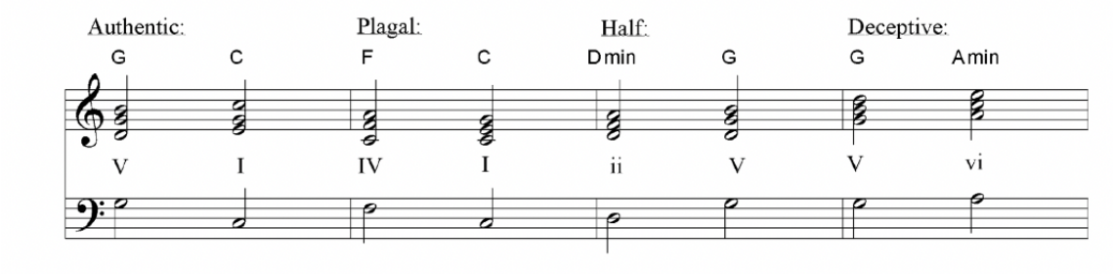

Figure 2.9: Cadences representation on music sheet.

Seen that this chord cadences can be found in the most common chords progression, the model use those element to decide which chords to use for a certain bar of the melody. The second factor taken in consideration by the model is the relationship between chords and melody. In particular, to have a chords progression coherent with the melody, chords which containing it are preferred. The next chapter is dedicated to describing those element and, more in general how the model works[24].

Transposition

Everything we have discussed up until now including the creation of the C /A scale the development of chords and cadences can be applied to other scales as well using transposition. Transposition is a core concept, in music theory that allows us to shift a passage or musical sequence from one key to another while preserving the structure and relationships, between the notes.

Chapter 3

Mathematical Model

3.1 The idea

Before proceeding with the analysis of the various components of the model, it is important to understand the underlying idea that guided its structuring. The goal of the structured model is to choose a sequence of chords that adhere to the rules of harmonization explained in the [music chapter] and, at the same time, are coherent with the melody provided as input by the user. Specifically, the aim is to select a number of chords equal to the number of beats entered by the user. Assigning a chord to each beat, in an amount equivalent to the entered beats, is a simplified way of achieving this objective. It is worth noting that a Mixed Integer programming model has been chosen to solve this type of problem. As mentioned earlier, the use of this solution is not very common in this research field; however, due to some similarities with the study conducted in the document "Generating guitar solos by integer programming" [1] this approach was chosen. In the cited article, a minimum path problem is solved, where each node in the graph represents a portion of a solo, referred to as a "lick." A weight is assigned to each edge connecting the "licks," influencing the model's choice in minimizing the objective function and selecting a series of nodes to create the complete solo. In the context of this model, the transition cost matrix plays a crucial role in the coherent selection of chord sequences. This matrix is not merely a series of numerical values but incorporates the valuable informational content derived from harmony theory. This theoretical knowledge enables the model to assess and choose chord sequences that are inherently coherent with the given melody. Harmonic rules and commonly used chord progressions in music theory, integrated into the cost matrix, serve as a guide for the model, facilitating the generation of accompaniments that adhere to the harmonic conventions of popular music. This fusion of theoretical knowledge and combinatorial optimization algorithms contributes to the model's ability to

create meaningful and musically satisfying harmonizations. Despite the peculiar difficulties inherent in the harmonization problem, a similar idea has been adopted to address it.

In the case of the melody harmonization:

- The nodes in the graph represent the chords of the accompaniment that could be chosen by the model;
- The edges represent the cost of transitioning from one chord to another.

In the cost decision-making process, it is crucial to carefully consider the parameters that influence the evaluation of transitions between chords. One of the main challenges is determining how to assign weights to different types of harmonic transitions. In particular, we will have two parameters with distinct functions, both of which, nevertheless, modify transition costs. One parameter will be dedicated to penalizing or favoring specific progressions, while the other will prioritize chords that better align with the melody in terms of the number of shared notes between the melody and the chord itself.

The flexibility of these parameters allows the model to adapt to various types of musical compositions and user preferences. Experimentation and optimization of these parameters are integral parts of the model's development journey. The main difficulty arises from the need to somehow connect these two elements to achieve a sequence of chords that is not only interesting but also coherent with the melody provided by the user.

3.2 Mixed Integer programming

This paragraph is useful to better understand the instrument that has been chosen to solve the problem of melody harmonization. Mixed Integer programming is a mathematical approach used to optimize a linear objective function subject to a set of linear constraints. In particular, LP involves making decisions about the allocation of resources to maximize or minimize a specific goal. The component of a Mixed Integer programming Model are the following:

- Decision Variables represent the quantities under consideration within the problem, exerting influence on the overall objective.
- The Objective Function is a linear equation which represent the goal, aiming either to maximize profits or minimize costs. It functions as the guiding metric in decision-making, reflecting the desired outcome.
- Constraints are relationships imposing restrictions on feasible values of decision variables.

The minimum path problem, often formulated as "find the shortest path between two points in a weighted graph," can be modeled through a system of linear equations, which is characteristic of LP problems. In this context, the objective function can be defined as the minimization of the sum of the edge weights along the selected path, and constraints can ensure the connectivity and consistency of the path. In the next paragraphs it will be explained how the minimum path problem has been used to model the problem of melody harmonization.

3.3 Variable

In this paragraph, we delve into the central aspect of the problem by defining the variables used to structure the model. Despite its complexity, the mathematical model developed to address the problem at hand involves only two key variables, each serving a specific role. Let's analyze these variables in detail, highlighting their significance and their contribution to solving the problem.

Binary Variables $x[i][j][t]$:

- If this variable has a value of 1, then the edge i —> j is included in the solution at time t.
- Conversely, it has a value of 0. This type of variable is essential for modeling the decision to select or exclude a specific edge at time "t".

Continuous Variables $p[i][j][t]$:

- The continuous variables p[i][j][t] represent the weight associated with the edge i —> j at time t.
- The use of variables structured in this manner allows for modeling the importance of edges in the graph dynamically, varying over time based on specific considerations. These weights reflect the "preference" for traversing a specific edge at time t, influenced by the user-inserted melody. Specifically, lower weights indicate better edges.

3.3.1 Time index

In the quest for a solution, each beat of the melody provided by the user necessitates the inclusion of a chord. This implies that, having chosen to use edges as variables in the model, there must be one for each transition from one beat to another. In the context of the mathematical model, the time index "t" represents a continuous time span connecting beat (or instant) k-1 to beat k. Instead of being a specific

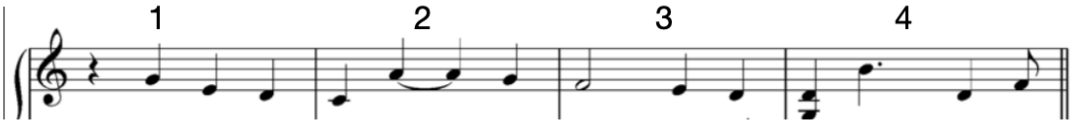

Figure 3.1: Four bar melody in the key of C major

moment in time, "t" delineates the temporal interval during which the transition from one beat to another occurs.

In the example shown in the figure, there are four beats. This might lead to the assumption that the number of instances of the variable "t" is equal to three. However, in reality, there are two additional edges necessary for modeling: the edge representing the transition from before the beginning of the piece to the first beat and the edge representing the transition from the last (fourth) beat to the graph's closure. This means that, in this case, "t" equals five. More generally, the number of instances of the variable "t" varies linearly with the number of beats "k" in the melody provided by the user according to the following equation:

$$t = k + 1 \tag{3.1}$$

3.3.2 Dummy Nodes

In the context of graph theory, a "dummy node" is a fictitious node introduced for specific purposes, such as simplifying the structure of a graph or facilitating certain operations. These nodes do not represent real elements of the system under study but are used to add structure or facilitate the calculation of optimal solutions.

Within the model, two additional dummy nodes have been introduced: the initial dummy node and the final dummy node. These nodes do not represent physical elements of the original graph but have been strategically included to facilitate mathematical modeling.

The initial dummy node serves to represent the concept that before the selected chord in the first beat, there are no other chords, but rather a dummy node representing the beginning of the accompaniment. It is connected to other nodes only through outgoing edges.

Similarly, the final dummy node represents the conclusion of the accompaniment and, in particular, the fact that after the selected chord in the last beat, there are no other chords. Unlike the "initial node," it is connected to other nodes in the graph only through incoming edges.

Considering the previous example:

- T = 0 represents the time span connecting the initial dummy node and the

first beat;

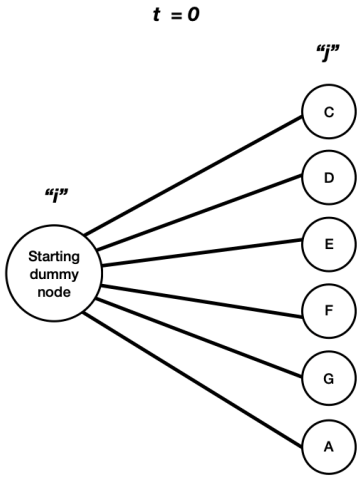

Figure 3.2: Fake starting node to start the graph

- T = 4 represents the time span connecting the last beat to the final dummy node.

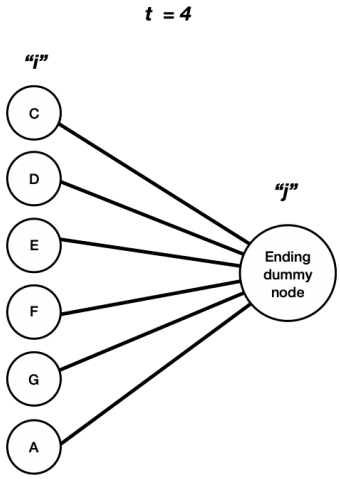

Figure 3.3: Fake starting node to start the graph

Once these dummy nodes are defined, the previously explained variables can be encapsulated in matrices. Specifically, it is not sufficient to define a single matrix "P" for weights and a matrix "X" with binary variables. For both mentioned variables, a number of matrices equal to k + 1 must be defined. This need arises from the fact that the edges connecting generic chords "i" and "j" should not have the same

weight regardless of the beats during the transition. Instead, the weights must change over time based on the melody provided by the user. The next paragraph presents the operations that modify these matrices before the model is solved.

3.4 Parameters

Before proceeding with the explanation of constraints and the objective function, it is essential to examine the preprocessing phase, a crucial step that adds a level of complexity to our approach. This adaptation of matrices, and so of the edges' weight, is fundamental to ensure that the model takes into account the specific characteristics of the melody, thereby contributing to generating chord sequences that organically integrate with the melodic progression. The operations that modify the cost of the matrix can be divided into two distinct operations:

- The first part of the preprocessing can be expressed as the incorporation of harmonic preferences into the model. The goal is to modify the weights of the edges to express a preference for the harmonic progressions explained earlier. It is important to emphasize that preferring a sequence of two chords i; j means assigning a lower cost to the edge connecting them, i.e., p[i][j][t]. In this phase, the modification of costs is independent of t and is carried out in the same way for each matrix. Next section will underline which arch weigth will be decreased.

  Chord matrix

  In general, when choosing a chord progression we aim for tensions and resolutions, and it is based on this general rule that the weights assigned to each arc in the graph have been determined. The aim of this paragraph is to show the chords data-set and to explain the weight that has been chosen for every arch between couple of chords. The table below shows all chords that have been used in this model, also explained in the paragraph "Constructing chord from the paragraph". To begin, each possible arch has been assigned a standardized weight of 50.

  The weights of the standard arches are then decreased or increased by a factor that is a parameter, called "bonus_chord_progression". This will be later take on different values to observe how its value influences harmonization. As the weight of an arch is increased, it becomes less likely to be included in the solution. Vice versa, for arches whose weight is decreased, they become more likely to be included in the solution.

  The following arches have been penalized:

Chord	Notes	Chord	Notes	Chord	Notes
C major	C - E - G	Cmaj7	C - E - G - B	C / E	E - C - E - G
D minor	D - F - A	Dmin7	D - F - A - C	C / G	G - C - E - G
E minor	E - G - B	Emin7	E - G - B - D	C / B	B - C - E - G
F major	F - A - C	Fmaj7	F - A - C - E	Am / G	G - A - C - E
G major	G - B - D	G7	G - B - D - F	Am / F	F - A - C - E
A minor	A - C - E	Amin7	A - C - E - G	E7	E - G# - B - D
B diminished	B - D - F	B half-diminished	B - D - F - A	D7	D - F# - A - C
				C7	C - E - G - Bb

Figure 3.4: Chords included in the model & their notes

- Arches towards semi-diminished and diminished B chords. In fact, the excessively unstable and tense nature of these chords makes it challenging to incorporate them into a tonal accompaniment. For this reason, a decision was made to penalize these arches. Nevertheless, it is still possible for the model to use these chords in cases where the melody specifically calls for them.
- Arches connecting identical chords but with different inversions have been penalized. This decision aims to prevent overly similar chords from being selected consecutively, avoiding repetitiveness and a lack of tension in the accompaniment.

To choose the arcs to prioritize through the weight bonus, we relied on the authentic, plagal, half, and deceptive cadences explained earlier. Additionally, the decision was influenced by the frequency of certain harmonic progressions. For this reason, the following arches have been favored:

- Arches which refers to cadences described before as Authentic and Plagal etc.
- Arches involved in the most common progression shown in the table below;
- Arches between secondary dominant chords and their I grade and the arch connecting the V grade of minor harmonic scale and A minor;

After this process, the matrix has significant differences with the starting one.

- The second part of pre-processing involves modifying the weights of chords based on the melody. To regulate this aspect, the decision was made to operate

Chord progression	Chords in C major
VI - IV - I - V	Am - F - C - G
VI - IV - V - I	Am - F - G - C
VI - V - IV - III7	Am - G - F - E7
VI - IV - I	Am - F - C
I - IV - V	C - F - G
I - V - VI - IV	C - G - Am -F
I - VI - IV - V	C - Am - F - G
I - VI - II - V	C - Am - Dm - G7
II - V - I	Dm7- G7 - C7+

Figure 3.5: Berklee College of Music's Website - Most common chords progressions [25]

as follows: the weight is decreased based on the number of notes in the melody that are also part of the analyzed chord.

An example is useful to explain this second modification to the matrix:

Suppose the user's melody consists of the notes C - E - F - D in the first beat, and let's consider the C major chord. As explained earlier, the edges connecting the initial dummy node to the chord played during the first beat belong to the matrix with t=0. The C major chord is represented by index j = 1, and the dummy node by index 0. Consequently, we modify the following edge: p[0][1][0]. In the proposed example, the melody notes also present in the C major chord, composed of the triad C - E - G, are C and E. The cost is then reduced by the following factor:

- $(Number - of - notes - in - common) * (MelodyBonus)$

The "Melody Bonus" is a quantitative indication of the preference given to chords containing a certain number of melody notes. It does not have a predefined value, as one of the objectives of this document is to analyze the various results provided by the software based on the modification of preprocessing parameters. To construct the matrix for use within the model, a matrix with equal weights is first created. Subsequently, these two operations are performed, and then the model is solved. It is important to underline that the preprocessing operations described are performed by setting parameters manually and are not automatic

operations. Essentially, the parameter related to the first preprocessing operation adjusts the importance of considering well-known cadences and chord progressions. The parameter concerning the melody, on the other hand, increases or decreases the fit between the chords the model chooses based on the melody. It is important to consider these two factors because incorrect settings can lead to inconsistent functioning of the model compared to the previously described musical rules.

3.5 Objective Function and Constraints

In this section, the key components of the optimization model, outlining both the objective function and the constraints will be analyzed. The objective function embodies the goal of the model. Instead constraints delineate the permissible solutions by imposing specific conditions. Together, the objective function and constraints guides the model towards generating optimal solutions within the defined problem space.

The objective function of this model is designed to minimize the overall cost associated with the selection of chords at different time instances. It is defined as the sum of the products of the binary variables $x[i][j][t]$ with the weights $p[i][j][t]$ for all edges i —> j and all times t. It is important to note that the matrix used is the one on which preprocessing operations have already been performed.

$$\min_{x_{ijt}} \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{t=1}^{k} x_{ijt} \cdot p_{ijt} \tag{3.2}$$

This optimization criterion aims to select a sequence of chords that minimizes the total cost of the path through the harmonic graph's edges, weighted by their respective preferences.

The following are the constraints to ensure a coherent solution:

The duration constraint limits the number of selectable edges in the solution, ensuring that only one edge is chosen for each time step t. Specifically, for each t, the summation of x over i and j is strictly equal to one.

$$\sum_{i=1}^{n} \sum_{j=1}^{m} x_{ijt} = 1 \quad \text{for each } t \tag{3.3}$$

The "Starting Constraint" ensures that the initial dummy node is chosen as the starting point of the graph. It is defined by the following equation:

$$\sum_{j=1}^{m} x_{0,j,0} = 1 \tag{3.4}$$

The requirement that the sum of all edges at t=0 with the starting node as the initial dummy node must be strictly equal to one, combined with the previous constraint, ensures that only one transition starts from dummy node "0."

The "Ending Constraint," similar to the starting constraint, ensures that the model's chosen path ends on the final dummy node.

$$\sum_{i=1}^{n} x_{i,n+1,n} = 1 \tag{3.5}$$

Let "n" be the total number of nodes, and "k" be the number of time instances. The equation defined here ensures that the only type of edge selectable at the last time instance is the one that terminates at the final dummy node.

The "Exclusion Constraint" defines a transition of chords that should not be chosen by the model:

$$\forall t \in \{1, 2, \ldots, k\}, \quad (i = j \implies x_{ijt} = 0) \tag{3.6}$$

In this specific case, the constraint prohibits the possibility of transitioning from a generic chord to itself at any time "t." This constraint has been added because, in the case of repetitive melodies, it prevents the same chord from being chosen for two or more consecutive beats.

The flow constraint ensures that for each intermediate node, the inflow is equal to the outflow, ensuring path connectivity. Simply put, if at t = k an edge to chord "j" is chosen, the next time instance must necessarily choose an edge starting from node "j."

$$\sum_{j=1}^{m} x_{ijt} - \sum_{i=1}^{n} x_{jit} = 0$$

This equation means that, if in a certain t, model choose the edge i ; j, in the next t, it has to choose and edge that is leaving from "j" to other nodes. On the other side, if the model does not choose i ; j as an arch, then no edges leaving from "j" can be used.

These constraints, along with the objective function, constitute the mixed integer programming model that is solved to obtain the optimal sequence of chords.

3.6 Next Steps in Model Development

To maintain a suitable level of complexity for the purpose of this document, various limitations have been introduced. The following are listed and explained:

- The model operates exclusively with the C major scale and its relative minor. This requires the user to transpose their melody to C, and after obtaining chords from the model, transpose them back to the original key. Even in the

testing chapter, this led to the selection of pieces only in this key. It could have been extended to work with other major scales, allowing the model to analyze melodies in different keys.

- The model is limited to working only with 4/4 time signatures. While complex rhythmic indications are excluded from our scope, the inability to use simple time signatures like 3/4, 2/4, 6/8, and 12/8 is limiting. These time signatures are common in various musical genres and could enhance the model's versatility.
- The model does not consider all possible harmonic solutions even within the C major scale, such as 9th, 11th, and 13th chords. Additionally, not all inversion chords are taken into account, only the most frequent ones in the context of light music. Including these would have provided more diverse harmonic possibilities.
- Modulations, including commonly used ones like "Parallel Modulation" and "Dominant Modulation," are not considered. Harmonizing melodies that do not remain in the same key throughout could be achieved by incorporating these modulation techniques.

All the mentioned limitations are retained because, with an increase in the number of chords, the complexity of managing the edges also escalates and so the complexity of handling edges' weight.

Chapter 4

WorkFlow and PseudoCode

4.1 Intro

To implement and solve the mathematical model discussed in this document, the translation into a practical and executable form was carried out using the Python programming language. The development process took place within the "PyCharm" integrated development environment (IDE), leveraging its functionalities for efficient coding and debugging. The realization of the model's logic and optimization was facilitated by the utilization of various Python libraries.

4.1.1 MIP

Firstly the Python MIP (Mixed-Integer Linear Programming) library served as a tool for formulating and solving the optimization model. This library provides a convenient interface for expressing the objective function, constraints, and transition costs involved in the model [26].

Key features of the Python MIP library include:

- Objective Function Formulation: The library allows the concise expression of the objective function, specifying whether the goal is to maximize or minimize a linear equation.
- Constraint Definition: Constraints, representing relationships and limitations on decision variables, can be easily defined using the library.
- Binary and Integer Variables: The MIP library supports the declaration of binary and integer variables, essential for handling decision variables $x_{ijt} = \begin{cases} 1 & \text{se l'arco (i, j) è nella soluzione al tempo t} \\ 0 & \text{viceversa} \end{cases}$

4.1.2 Gurobi

MIP itself does not include solvers; instead, it interfaces with external solvers to solve optimization problems. For the actual solution of the formulated optimization problem, the Gurobi solver was employed. Gurobi is a state-of-the-art optimization solver renowned for its efficiency and performance, particularly in solving linear programming and mixed-integer linear programming problems [**<empty citation>**].

Key characteristics of the Gurobi solver include:

- Optimization Algorithms: Gurobi incorporates advanced optimization algorithms to efficiently explore the solution space and identify optimal solutions.
- Scalability: Known for its scalability, Gurobi is capable of handling large-scale linear programming problems with a vast number of decision variables and constraints.
- Python Interface: Gurobi seamlessly integrates with Python, enabling the use of Python scripts to interact with and solve optimization problems.

4.1.3 Music21

The software, aiming to receive a melody as input and subsequently generate a musical accompaniment, requires a means to handle musical files both in input and output. Specifically, the software utilizes sheet music written in the musicXML format. To read and manipulate elements of the harmonization, such as sequences of notes and chords, the music21 library has been employed [2].

The music21 library is a versatile Python tool designed for the manipulation and analysis of musical data. It enables the reading, writing, and manipulation of musical scores in musicXML format. Music21 provides advanced functionalities for extracting musical information, such as notes, chords, melodies, and more, making it the ideal choice for the analysis and management of musical data within the context of our software.

4.2 Workflow and Pseudo-Code

The following description outlines the program's workflow through the use of pseudocode. This approach allows for conveying the essence of the algorithm without being constrained by a specific programming language or particular libraries. It is worth noting that the representation through pseudocode is an intentional choice to provide a high-level overview independent of the actual implementation.

Pseudocode is an informal programming language that provides a clear and understandable way to express the algorithm without specifying the syntax of a

particular programming language. This facilitates an understanding of the workflow without being influenced by implementation details. In the next section, a brief description of the workflow using pseudocode will follow, enabling a universal understanding of the process regardless of the programming language used for practical implementation.

The program's operations associated with the pseudocode for their implementation are listed below:

1. To begin the process, the user is required to provide two essential pieces of information. Firstly, the software needs the melody file that the user intends to include. Secondly, the user must specify the number of measures, denoted as "n," constituting the given melody. In particular, at the line "8", the melody number of bar is asked to the user.

Algorithm 1 Input del percorso del file MusicXML

```
1: Input: file_path (string)
2:
3:                          ▷ Request user to input the path of the MusicXML file
4: file_path ← input("Enter the path of the MusicXML file:  ")
5: Input: None
6: Output: num_bars (integer)
7:
8: num_bars_input ← input("Enter the number of measures:  ")
9: num_bars ← convert_to_integer(num_bars_input)
```

2. The software, at this point, knowing this information, should extract from the file a list of "n" measures that compose the melody. Each element of the list contains the notes of the nth measure. More precisely, on line number 2, the first bar of the melody provided by the user is extracted. Subsequently, on line 4, a list of bars is initialized, the length of which is based on the previously entered number by the user. Starting from line number 5, an iteration is performed through the bars of the melody. For each bar, starting from line 6, there is an iteration over the notes belonging to that measure. On line 7, each note is added to the previously created list. In lines 10 and 11, there is an iteration over the list to print the notes belonging to the melody.

Algorithm 2 Read and Print Melody Measures

```
function ReadAndPrint(file_path, num_measures) ▷ Load the MusicXML file
    score ← parse(file_path)
    part ← score.parts[0]
    measures ← [[] for _ in range(num_measures)]
    for measure ∈ part.getElementsByClass(stream.Measure)[:num_measures] do
        for note ∈ measure.flatten().notes do
            measures[measure.number - 1].extend(str(p) for p in note.pitches)
        end for
    end for
    for i ← 1 to num_measures do
        Print("Measure", i: measures[i − 1])
    end for
     return measures
end function
```

3. A total of $n + 1$ matrices are now created to represent the transition costs from chord i to chord j for each time instance. In this initial phase, the transition costs are all set to the same value. It's important to remember that, in addition to the chords intended to be inserted as nodes in the graph, the initial dummy node and the final dummy node must be added. The variable "standard cost" is a global variable which represent the initial cost of all the edges. In this way, it will be possible to change that value at any time to understand the differences in the model's behavior as this parameter varies.

Algorithm 3 Initialization of Matrix p

```
p ← [[[standard_cost for t in range(k)] for j in range(m)] for i in range(n)]
```

4. The first preprocessing operation occurs at this point: there is a need to modify the transition costs of all matrices to make certain chord successions more or less preferable. The implementation of this phase depends strictly on the encoding between nodes and chords that has been chosen. The first step, in the first line, is to iterate over all matrices related to various time instances defined by different bars. The bonus_progression mentioned in lines 2 and 3 is the parameter that determines how certain edges are favored or penalized. In the second line, the value of the edge is decremented, favoring its selection in the model's output volume. On the other hand, in the third line, this value

is incremented, penalizing the selection of the edge within the graph.

Algorithm 4 Preprocessing for Transition Costs

```
for t in range(k) do
    p[i'][j'][t]− = bonus_progression
    p[i'][j'][t]+ = bonus_progression
end for
```

It is important to underline that if we increase the value, we are penalizing a transition, and vice versa.

5. The second preprocessing operation, as seen earlier, involves preferring edges $i \to j$ in which the destination chord (j) contains more notes than the melody. A comparison between the notes of a certain measure and the notes contained in chord j is made. For each common note, the weight of the edges ending in j at that time instance is reduced. The same operation is extended to all measures and all chords.

Algorithm 5 Modify Cost Matrix

```
function ModifyCostMatrix(p, k, n, m, reduction_cost, list_of_beats)
    Input: p (matrix), k (integer), n (integer), m (integer), reduction_cost
(float), list_of_beats (list)
    length ← Length(list_of_beats)
    for t ← 0 to k − 1 do
        for l ← 0 to length − 1 do
            if t == l then
                for j ← 0 to m − 2 do
                    if j == 1 then
                        do ← MajorChord()                    ▷ Create major chord
                        count ← CountNotesInChord(do, list_of_beats[l])
                        for i ← 0 to n − 2 do
                            p[i][j][t] −= count × reduction_cost
                        end for
                    end if
                end for
            end if
        end for
    end for
end function
```

6. At this point, the model is defined and solved by implementing the objective

function and constraints on the nodes. The solution is the path between nodes (chords) that minimizes the cost of the objective function. In particular, at third line, the objective function is established to minimize the sum of products between decision variables (x) and their corresponding weights (p) across time intervals, nodes, and tasks. Then, on lines 5-6, constraints are introduced to ensure that the path starts from an initial dummy node (node 0). Specifically, the sum of edges leaving this node in the first time interval (t=0) is set to be equal to 1, indicating that exactly one edge must leave the initial node. Similarly, constraints are imposed to ensure that the path reaches the ending dummy node (node m-1) in the last time interval (t=k-1). The sum of edges reaching this node is set to be equal to 1, ensuring that exactly one edge must reach the ending node. Lines 14 to 17 use loops and conditional statements to set constraints preventing the selection of edges with the same starting and ending node (i == j). If this condition is met, the corresponding decision variable is forced to be 0. On lines 21-25 flow constraint is set, ensuring the conservation of flow between time intervals (t). The "flow constraint" ensures that if flow arrives at node j at a certain time instance, in the subsequent time instance, it is mandatory to depart from that specific node. This constraint reflects the continuity of flow within a network, emphasizing that the flow into a node in a given time interval should equal the flow out of that node in the subsequent time interval. This condition maintains a coherent and balanced flow pattern throughout the optimization model, contributing to the overall effectiveness and realism of the solution.

Algorithm 6 Objective and Constraints

Objective:
$model$.objective $\leftarrow$ minimize $\left(\sum_{t=0}^{k-1}\sum_{i=0}^{n-1}\sum_{j=0}^{m-1} x[i][j][t] \cdot p[i][j][t]\right)$
Constraints:Start on the initial dummy node
$model+ = \sum_{j=1}^{m-1} x[0][j][0] == 1$
Constraints:Finish on the ending dummy node
$model+ = \sum_{i=0}^{n-1} x[i][m-1][k-1] == 1$
for $t = 0$ **to** $k-1$ **do**
    $model+ = \sum_{i=0}^{n-1}\sum_{j=0}^{m-1} x[i][j][t] == 1$
end for
Constraints:Avoid edges with the same starting and ending chord
for $i = 0$ **to** $n-1$ **do**
    for $j = 0$ **to** $m-1$ **do**
        for $t = 0$ **to** $k-1$ **do**
            if $i == j$ **then**
                $model+ = x[i][j][t] == 0$
            end if
        end for
    end for
end for
Flow constraint
for $t = 1$ **to** $k-1$ **do**
    for $i = 0$ **to** $n-1$ **do**
        $model+ = \sum_{j=0}^{m-1} x[i][j][t] == \sum_{j=0}^{m-1} x[j][i][t-1]$
    end for
end for

7. The part of code showed below is designed to display the optimal solution of a mathematical optimization model. Firstly, at lines 2-3, the algorithm creates an empty list called "solution". Between lines 4 and 19, the algorithm examines the optimization model (denoted as "model") to determine if any solutions have been identified, as confirmed by the condition "model.num_solutions $>$ 0." In the event of a positive outcome, the algorithm initiates a set of nested loops, starting with the iteration over time intervals (t) from 0 to k-1. Within this temporal loop, additional nested loops iterate over resources (i) and tasks (j). For each specific combination of i, j, and t, the algorithm assesses the decision variable x[i][j][t] within the solution. If the value surpasses or equals 0.99, it designates the task as part of the optimal solution. During this process, the algorithm prints a message conveying the scheduling information, specifying that task j is allocated at time t for resource i. Simultaneously, the task index

j is appended to the "solution" list, which serves as a dynamic record of the optimal solution's composition. At line 20, the algorithm removes the last element from the "solution" list (using solution.pop()); in fact the last dummy node is not a chord for the accompaniment, but just an artefact needed for the model.

Algorithm 7 Display Optimal Solution

```
1: Output:
2: solution ← []
3: if model.num_solutions then
4:     Print("Optimal solution found:")
5:     for t ← 0 to k − 1 do
6:         Print("Time interval", t + 1:)
7:         for i ← 0 to n − 1 do
8:             for j ← 0 to m − 1 do
9:                 if x[i][j][t].x ≥ 0.99 then
10:                    Print("x_i_j_t = 1")
11:                    Append j to solution
12:                end if
13:            end for
14:        end for
15:    end for
16:    Print("Objective function value:", model.objective_value)
17: else
18:    Print("No optimal solution found.")
19: end if
20: solution.pop()
21: Print(solution)
```

This pseudocode represents the logic for displaying the optimal solution found by the model.

8. The final step is to return to the user a file that combines the initially input melody and the chords chosen by the model.

float

Chapter 5

Testing and Results

5.1 Introduction to testing

Testing a model for melodic harmonization poses a unique challenge, as evaluating the model's choices cannot rely on objective criteria. The musical interpretation of the generated accompaniment is inherently subjective, varying from person to person based on individual preferences and musical experiences. The complexity of melodic harmonization further complicates the establishment of universal evaluation standards.

The generated accompaniment may be perceived as excessively repetitive by one listener, while another might find it characterized by too many tensions and resolutions. To address this challenge, we have chosen an approach involving well-known musical pieces. We extract the melody from these songs, represented by the vocal line, and subject it to our melodic harmonization model. Throughout this process, we vary the model's parameters to explore its behavior and understand how such variations impact the chosen harmonization.

The goal is to compare the harmonization proposed by the model (with varying parameters) with the original accompaniment of the reference song, possibly identifying which set of parameters brings the model closest to the original piece. Through this comparison, we aim to analyze differences, identify potential errors, and pinpoint similarities. While the evaluation remains somewhat subjective, this approach provides a more in-depth perspective on the model's capabilities within the context of well-known musical pieces.

5.1.1 Song selection and input data analysis

To understand the choice of songs for testing, it's crucial to recall the limitations of the model explained in the preceding chapters. Specifically, constraints such

as the requirement for a 4/4 time signature and adherence to the C major scale compelled us to select songs that align with these characteristics.

Furthermore, to conduct a meaningful comparison, we had to choose songs that, much like the arrangements generated by the model, feature only one chord per bar. Moreover, the absence of certain harmonic elements like 9th, 11th, and 13th chords, as well as more complex modulations, narrows the selection to songs that aren't overly intricate in terms of accompaniment. The three songs chosen for the test are:

- Someone Like You - Adele
- My Way - Frank Sinatra
- I will survive - Gloria Gaynor

The sheet music for these songs was downloaded from the MuseScore website, from which we also obtained the software to handle the XML files used as input/output for the model. At this point, one final step was required: the accompaniment part was removed from the sheet music, leaving only the melody.

Before the results of the tests are discussed, it is useful to note that light music tracks follow the same chord progressions throughout their duration. Our model, on the other hand, selects chord progressions based on the melody's progression and, consequently, tends not to repeat chosen chord patterns. That being said, we cannot expect the solution proposed by the software to be a sequence of chords identical to that of the original song. However, with the right parameters, it should be able to generate a coherent chord progression that complements the song.

5.1.2 Testing parameters

As mentioned earlier, various parameter settings of the model will be used to conduct the test. As stated before, the edge weights start from a standardized value, but subsequently, with the pre-processing operations, they are modified based on two parameters:

- *bonus_progression* contributing to penalize or favor an edge based on the predictability of the chord progression it represents;
- *bonus_notes* influencing the weight of the edge based on the number of melody notes that are in common with the chord.

In the table below, the pairs of parameters chosen to conduct the test on previously tested songs appear.

5.2 1th song testing - Someone like You

In order to compare the results of our software testing, it is essential to introduce the original harmonic structure of the song 'Someone Like You.' The harmonic framework is comprised of verses and choruses featuring the chords C Major, E Minor, A Minor, and F Major; this chord progression is showed in the next figure 5.1.

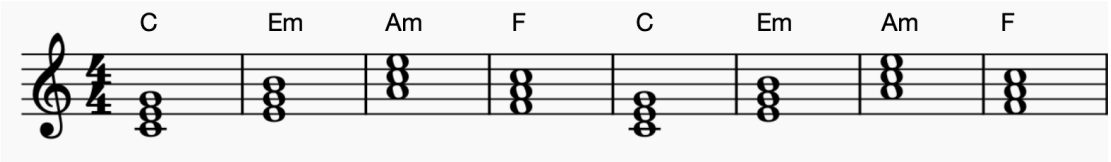

Figure 5.1: Someone like you - Original chord progression

Before delving into the analysis of each parameter set individually, let's first review this table summarizing the results, which will then be examined in detail."

Original chords Progressions	C - Em - Am - F
Set 1 - 1st stanza	E7 - Am7 - F - E7
Set 1 - 2nd stanza	Am7 - Am - F - E7
Set 2 - 1st stanza	C/E -C/G- C/E - C/G
Set 2 - 2nd stanza	C/E - Dm7 - Am7 - C/E
Set 3 - 1st stanza	C - Am - C/G - Em
Set 3 - 2nd stanza	C/G - Dm7 - Am7 - C
Set 4 - 1st stanza	C - Am -C/E - Dm7
Set 4 - 2nd stanza	C/G- Em -C/E - Dm7

Figure 5.2: Summary of model performance on 'Someone Like You'

Set 1

The first set combination we are evaluating includes a *bonus_progression* set to 15 and a *bonus_note* set to 5. Already from the weights, it can be inferred that this combination is somewhat skewed towards the parameter related to chord progression rather than the one concerning the melody.

The model's generated sequence introduces variations in chords, incorporating E dominant 7th, A minor 7th, F Major and D Dominant 7th. Notably, the model exhibits a preference for certain chord progressions, particularly the E7 - Am (V-I) cadence, which is consistently chosen. Additionally, the transition from A minor to F Major (Am - F) is a recurring choice in both sequences, underscoring its prevalence in pop chord progressions. This aligns with the model's tendency to favor common progressions while strategically avoiding penalized transitions, as reflected by its lower weights in the model structure. The effect on the model's output caused by this set of paramaters can be observed in the figure 5.3 In this

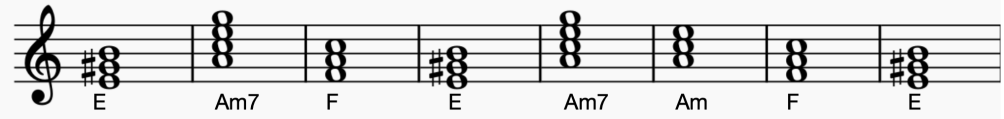

Figure 5.3: Chord progression chosen by the model for Someone Like You - Set 1

manner, for the song 'Someone Like You,' the model generates chord progressions that are not at all coherent with the analyzed piece. This occurs because the bonus assigned to the progression carries an excessively higher weight compared to the one associated with the melody notes. As a result, the chord progression proceeds without considering the melody, resulting in a complete disconnect between the two. Of course, the same result is observed when the *bonus_progression* is further increased, even with a proportional increase in the second parameter. "And vice versa, keeping the first parameter constant and decreasing the *bonus_note*.

5.2.1 Set 2

The second set combination under evaluation involves a *bonus_progression* set to 5 and a *bonus_note* set to 15. In contrast to parameter set number 1, here the imbalance is towards the bonus related to the melody notes. It is foreseeable that in this case, the model will tend to select chords that are too similar to each other sequentially. This occurs because, typically, the melody of a song does not change significantly from one verse to another, with only small variations. Consequently, the model consistently chooses the same chords, reflecting this characteristic pattern in its output.

The model tends to generate chords that are almost identical in succession, particularly sequences like C/E, C/G, C/E, which we intended to penalize by assigning increased weights to corresponding edges through the *bonus_progression*. However, due to the considerably higher weight assigned to the other parameter, the penalization applied to certain progressions has little to no effect. In the figure 5.4, it can be observed the result, in terms of chords produced by the model on a music sheet. The model tends to generate chords that are almost identical in succession, particularly sequences like C/E, C/G, C/E, which we intended to penalize by assigning increased weights to corresponding edges through the *bonus_progression*. However, due to the considerably higher weight assigned to the other parameter, the penalization applied to certain progressions has little to no effect. We can assert that when the melody parameter outweighs the progression

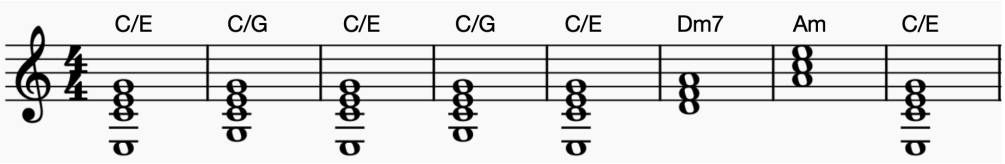

Figure 5.4: Chord progression chosen by the model for Someone Like You - Set 2

parameter, the model generates a series of repetitive chords, closely following the melody without considering the 'rules' regarding cadences that we intend it to adhere to in the progression.

5.2.2 Set 3

This set of parameters, in comparison to the previous ones, has balanced weights.It is predictable that this setting will lead to a more balanced chord progression compared to before. Upon conducting an actual comparison between the chords produced by the model and the original ones, we observe that the chord progression produced by the model in start to be similar the the one of original chords. In particular, in the original accompaniment is composed by the following chords progression repeated over and over in verses and chorus :

- C - Em - Am - F;

. The first thing to notice is that, every four bar, as in the song, the model uses C major chords or its inversion as first of the four chords progression.This can be observed in the first two stanzas represented on the music sheet in figure 5.5 Despite the slight differences in how inversion chords sound, it can still be affirmed that the first chord of the progression is consistently chosen correctly, except for this minor difference in the bass note of the chord.

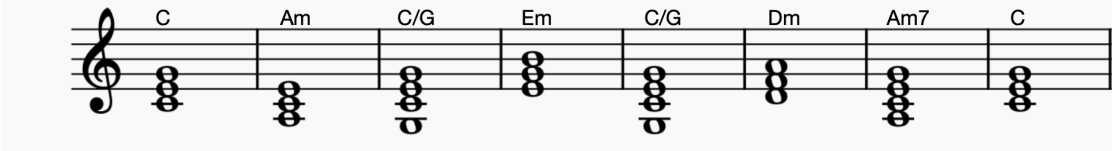

Figure 5.5: Chord progression chosen by the model for Someone Like You - Set 3

As for the third chord of the progression, which in the original song is always A minor, we can observe that the model replaces it with C/G, A minor 7, C/E. It is important to note that, even though the A minor 7 chord adds a richer harmonic color, consisting of the same notes as the A minor chord (A - C - E) with the addition of the 7th degree (G), it can still be considered suitable. A similar consideration applies to the C/G chord, which in one instance replaces the A minor in the original progression: the C/G chord is composed of the notes C - E - G, while the A minor is formed by A - C - E. Despite these two chords being different, they are interchangeable within a song, making this substitution satisfactory as well. However, it is worth noting that a challenge still encountered with these parameter values is that, at certain points, the chosen chords closely follow the melody and fail to avoid, as intended, the chord transitions that were meant to be penalized. Specifically, it still occurs that more than two inversions of the same chord are played consecutively, such as C/G, C/E, C/G. In addition to not being able to recreate an arrangement similar to the original, this poses the issue of following a progression that, in the model's construction, we aimed to avoid precisely due to its repetitive and monotonous nature.

5.2.3 Set 4

Among the settings chosen for a more in-depth analysis, this combination of parameters appears to be the best at balancing the factor related to the melody and that concerning chord progressions. As seen in the previous pair of parameter values, in this case as well, the first and third chords of the original progression are correctly replaced within the model by using chords that are very similar to the originals. In particular, the C major chord is replaced by some inversions of the same C major, and the A minor is replaced by A minor 7 or by inversions of C that still share 2 or more notes with A minor. In addition, in two measures, the F major, which should be the last chord of the progression, is effectively replaced by its relative minor, D minor 7. In fact, the F major is composed of the notes F - A - C, while the D minor 7 consists of D - F - A - C. Having 3 notes in common, these chords are interchangeable within a progression. This can be observed in the music sheet produced by the model for the first two stanzas in the figure 5.6

The issue lies in the fact that, in the model's progression, there are some

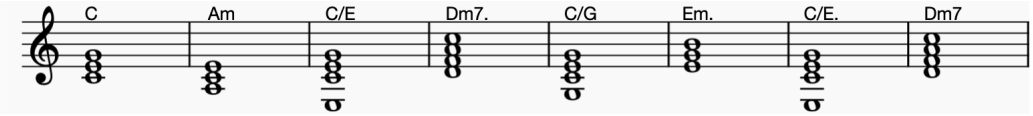

Figure 5.6: Chord progression chosen by the model for Someone Like You - Set 3

variations compared to the original sequence of the song. In particular, the model introduces additional chords such as E Dominant 7th and D Dominant 7th. Although these chord variations enrich the variety of harmonic colors within the song, they do not exactly mimic the cadences of the original composition.

5.3 2nd song testing - My Way

The second song chosen for testing is "My Way" by Frank Sinatra; compared to the previous song, it features a more complex harmonic structure.This chords progression, is showed on a musical sheet in the figure 5.7.

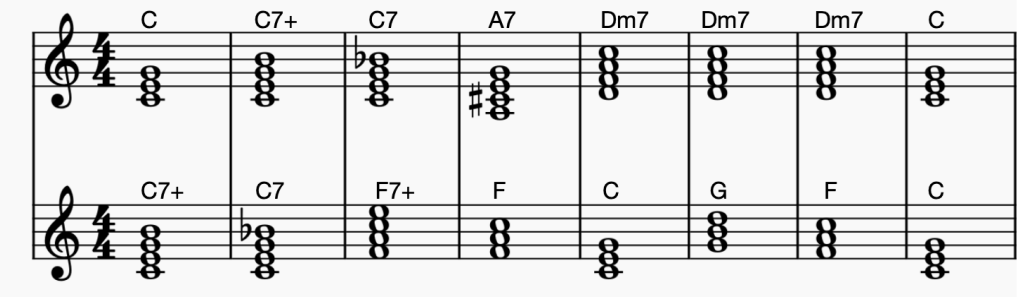

Figure 5.7: My Way - Original chord progression

In particular, the song begins with a harmonic descent of the octave of C major, which then becomes a seventh and then a minor seventh with the progression C major, Cmaj7, and C dominant 7. Subsequently, there is a secondary dominant substitution with the use of A major 7. This A7 chord is followed by a D minor 7 that imitates the authentic 5-1 cadence like the authentic cadence of the harmonic minor scale. Then, it moves to G7 (V7), resolving to C major, and repeats the progression C major, Cmaj7, and C7. The last part of the verse recalls cadences explained in chapter two, specifically F - C (plagal), G - F (deceptive), and again F - C.

Before delving into the analysis of each parameter set individually, let's first review this table summarizing the results, which will then be examined in detail.

Original chords Progressions	C - CMaj7- C7 - A7 - Dm7 - Dm7 - Dm7 - C CMaj7 - C7 - F7 - F - C - G - F - C
Set 1 - 1st stanza	G - F - E7 - Am7 - G - E7 - Am - F E7 - Am - G - Am7 - F - E7 - G - F
Set 2 - 1st stanza	G - C/G - C/E - Em7 - C/E - A/F# - A/F - F G/B - C/E - C/G - Am7 - A/F - G/B - C/E - G
Set 3 - 1st stanza	C/E - Em7 - C/E - A/F# - A/F - F7 - G7 - C/E C/G - A/G - F7 - G7 - C/E - G - Dm - C/G
Set 4 - 1st stanza	C/E - Em7 - C/E - A/F# - A/F - F7 - G7 - F7 C/G - Am7 - F - G/B - C/E - G - F - C

Figure 5.8: Summary of model performance on 'My Way'

Set 1

In the first set of parameters, the *bonus_progression* has a value of 15, and the *bonus_note* has a value of 5. As previously observed, analyzing the first piece revealed that this combination leads to an imbalanced behavior of the model. In the case of this piece, the effect is almost the same. The model generates the following chord pairs:

- E dominant 7 and A minor, as we have seen before, represent the authentic cadence from the fifth to the first degree of the harmonic minor scale.
- F major to C major, representing the plagal cadence of the C major scale.
- Lastly, the progression from A minor to F major, a chord pair frequently used in light music, which was favored within the model.

Essentially, the model in this case randomly selects chord pairs that, when played in succession, may sound acceptable, but they lack coherence with the given melody.The behavior of the model is clearly visible in the music sheet it produces with the aforementioned parameters.

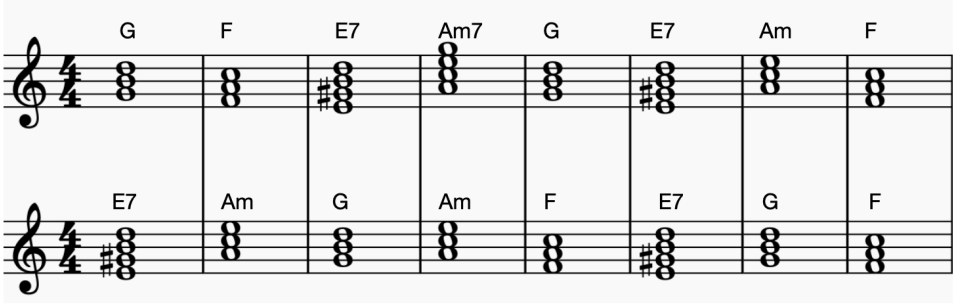

Figure 5.9: Chords progression chosen by the model - Set 1

This happens because of a strong imbalance in weights, where the parameter intended to allow the model to "adapt" the accompaniment to the specific melody has no effect. It is worth noting that, due to the reasons described, the generated chords differ from those used in the original piece.

Set 2

The parameter set described in this section is presented as the opposite of set number 1; similarly to it, the weight set is skewed towards the bonus note. What happens in this case is that the chord progression closely follows the input melody. In the first part, instead of selecting the cadence C, Cmaj7, and C7 as in the original piece, the model opts for two inversions of C major and a Dm7. The inversions of C major can be considered accurate for the model, while Dm7 is chosen because, in the second measure, the melody notes precisely match those of this chord.

Subsequently, in the original piece, an Amaj7 is used as a secondary dominant, and the model manages to predict this, at least with these parameters. In the original, Amaj7, being a dominant chord, is followed by a Dm7 to create an authentic cadence. However, the model selects an Fmaj7 instead. This chord is the relative major of the Dm7 chord, and there isn't much difference in terms of notes between them. Hence, they can be interchangeable within a progression.

Later on, the original piece shifts to a G7 dominant chord and then resolves to Cmaj7. Our model does the same but starting from the next measure. By adhering too closely to the melody, it deviates from the original chords of the piece from this point onwards.

Unlike what happened with the first piece, this proportion of parameter values

works better here. In particular, it is almost non-repetitive, and there are no situations where similar chords are repeated more than twice. We can infer that, in this case, the melody is less repetitive, and consequently, so are the model's choices.

The figure 5.10 below, shows the musical sheet produced by the model when using the second set of parameter.

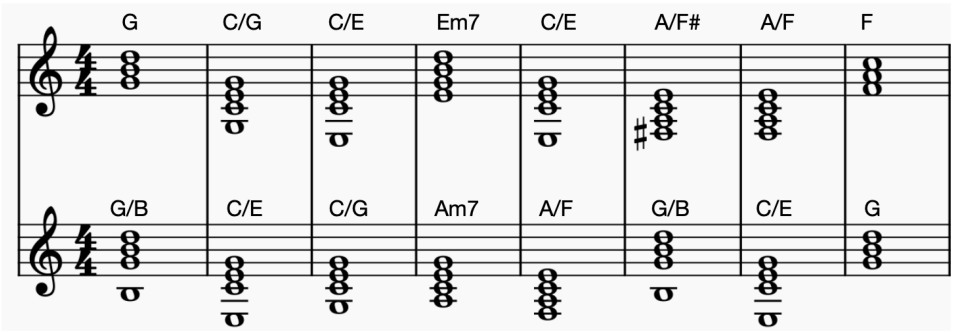

Figure 5.10: Chords progression chosen by the model - Set 2

Set 3

The parameter set number three assigns the same value to both *bonus_progression* and *bonus_note*, making it defined as balanced. However, in this piece, the behavior is not significantly different from parameter set number two.

Analyzing the chords generated by the model, it is observed that throughout the first stanza, they are the same as those produced by set 2. More specifically, two of the first three chords of C major from the original piece are correctly generated. Additionally, here too, the model identifies the presence of a secondary dominant and an authentic cadence at the end of the first stanza.

Moreover, compared to the previous set, the model successfully imitates the ending of the second stanza faithfully: the original piece features a sequence of the fifth and fourth degrees, namely, G7 dominant and F major, followed by the plagal cadence from F major to C major to close the stanza. Similarly, the model generates the Gmaj7 dominant chord followed by Dm7; recall that the latter is the relative minor of the F major chord and can be considered interchangeable with it. Subsequently, the C major chord is generated to close the stanza in the same way as the original. As happened in the first tested piece, here too, the balance of

weights makes the model more effective in generating chords more similar to the original ones. In the score showed in figure 5.11, can be observed more similarity with the onw in figure 5.7

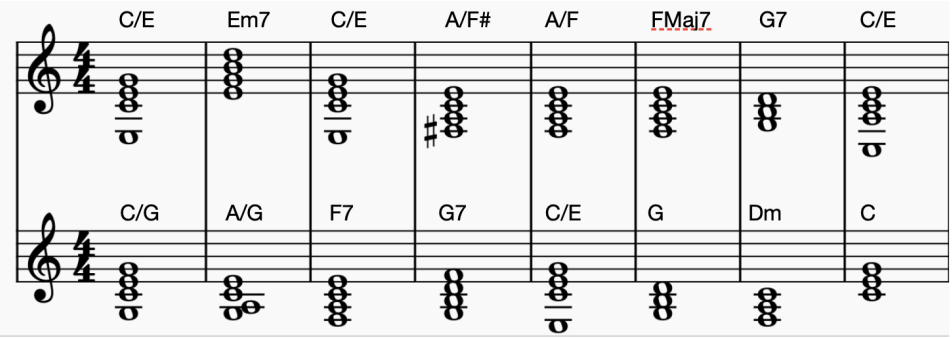

Figure 5.11: Chords progression chosen by the model - Set 3

Set 4

In the model test on this piece, parameter set number 4 is the one that generated chords most similar to those belonging to the original accompaniment. Specifically, regarding the first stanza, it achieves the same results as sets 2 and 3, predicting only two of the first 3 chords of C major and then using the secondary dominant. It is worth noting that, to avoid overly repetitive progressions, similar chord progressions were penalized during the preprocessing stage. This operation ensures that the model does not choose three consecutive C major chords but inserts another chord to avoid overly frequent transitions, intentionally made more costly.

As for the parameter set before, in the final part of the second stanza, it utilizes the following progression: G major, D minor, and C major. Unlike the original piece, which has F major instead of D minor, parameter set number 4 eliminates this difference between the model-generated accompaniment and the original one. In fact, it exactly generates the sequence of the mentioned final 3 chords. This can be clearly seen observing the original score in figure 5.7 and the one produced by the model in figure 5.12. Additionally, even the chord preceding this progression, namely C major, is correctly generated by the model.

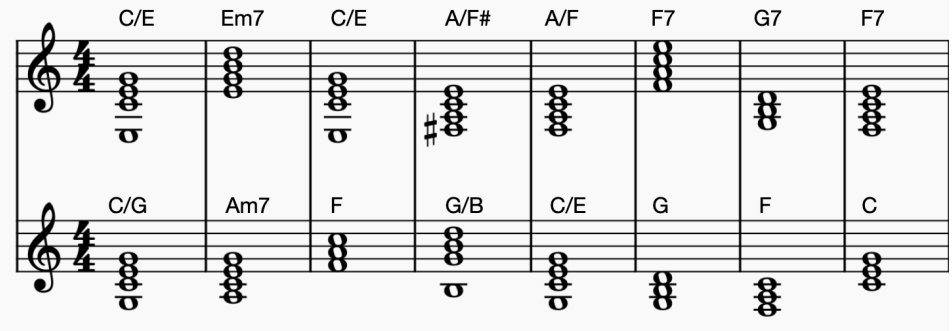

Figure 5.12: Chords progression chosen by the model - Set 4

5.4 3rd song testing - I will Survive

The last song chosen for the testing phase is "I Will Survive" by Gloria Gaynor. It is essentially formed by a eight bar chord progression in the key of A minor, which is relatively more complex compared to "Someone Like You". This structure repeats cyclically throughout the duration of the song. In particular, the song presents the following chord sequence: Am - Dm - G7 - Cmaj7 - Fmaj7 - Bm7b5 - E7 - E7. This chord progression is represented on a music sheet in the figure 5.13; at the end of this sequence, it always returns to the initial Am chord. It's important to note that we're analyzing only the first stanzas and not the entire piece. This is because, given the cyclic nature of the melody and harmonic structure every 8 beats, the model consistently generates the same progression from a certain point onwards. We can observe that the authentic cadence formed by the succession of

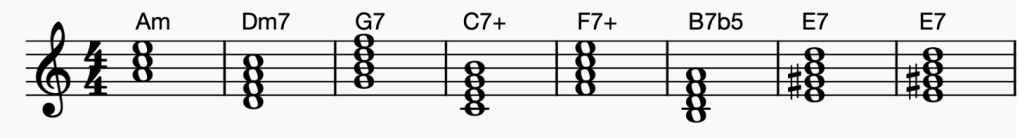

Figure 5.13: I Will Survive Chord Progression

the V chord and the I chord is used multiple times. Between the third and fourth chords of the progression, there is a transition from G7 to Cmaj7, which are the

fifth and first degrees of the C major scale respectively; furthermore, the last chord, E7, represents the fifth degree of an A harmonic minor scale and is followed by an A minor chord, which in this context functions as the tonic (I) chord. Prior to delving into the individual analysis of each parameter set, let's begin by reviewing this table 5.14summarizing the results. Subsequently, we will proceed to examine them in greater detail. Before proceeding, it's important to note that in the sheet

Original chords Progressions	Am - Dm - G7 - Cmaj7 - Fmaj7 - Bm7b5 - E7 - E7
Set 1 - 1st stanza	Am7 - F - C/G - Em - C/E - Em7 - C/E - Emaj7
Set 1 - 2nd stanza	Am7 - F - C/G - Am7 - C/E - Emaj7 - C/E - Emaj7
Set 2 - 1st stanza	Am7 - Dm - C/G - C/E - Emaj7 - Emaj - C/E - Emaj
Set 2 - 2nd stanza	Am7 - Dm - C/G - Am7 - C/E - Emaj7 - C/E - Emaj7
Set 3 - 1st stanza	Am - Dm7 - C/G - C/E - Emaj7 - Em7 - C/E - Emaj7
Set 3 - 2nd stanza	Am7 - F - C/G - Am7 - C/E - Emaj7 - C/E - Emaj7
Set 4 - 1st stanza	Am7 - F - C/G - C/E - Fmaj7 - Em7 - C/E - Emaj7
Set 4 - 2nd stanza	Am7 - F - C/G - Am7 - C/E - Em7 - C/E - Emaj7

Figure 5.14: Summary of model performance on 'I Will Survive'

music images representing the model's results for the first stanza, two empty bars are consistently observed. This occurs because at the beginning of the piece, there are two introductory bars with an E7 arpeggio that lacks accompaniment, resulting in the first two bars having no chords.

Set 1

Using the first set of parameters on this excerpt with the *bonus_progression* having a value of 15, and the *bonus_note* having a value of 5, the result is not as unbalanced as in other cases. In particular, the model chooses the following chords for the first stanza:

- A Minor 7 - F Major - C/G - E Minor - C/E - E Minor 7 - C/G - E Major 7 ;

the result about the first stanza is represented in the figure 5.15

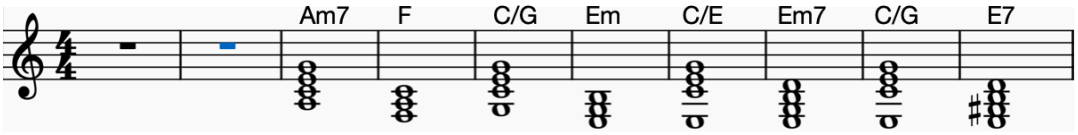

Figure 5.15: Chords progression chosen by the model for the first stanza

Firstly, we notice that the model correctly selects the first chord of the progression, which is A minor 7. The same can be observed regarding the last chord of the progression, which is E major 7. Additionally, we recall that in the original progression, after the initial A minor, there is a D minor 7; in this case, the model selects F major instead. As seen previously, D minor 7 consists of the notes:

- D - F - A - C ;

F major is also formed by 3 of these notes, namely F, A, and C; consequently, we can consider this substitution correct from a harmonic and coherence standpoint of the piece. Regarding the central part of the harmonization, however, this set of parameters produces different chords compared to the original. The too high weight of the parameter concerning progressions causes the model to often choose the same cadences, for example, between the third and fourth chord, fifth and sixth, seventh and eighth. In the figure 5.16 this problem can be clearly seen and those below are the chords produced by the model:

- C/G - E minor
- C/E - E minor 7
- C/E - E major 7

This behavior becomes even more evident when observing the second stanza produced by the model:

- A Minor 7 - F Major - C/G - A Minor 7 - C/E - E Major 7 - C/E - E Major 7;

We can thus conclude that even in this case **??**, the parameters of the first set produce a result that tends not to follow the melody of the piece but rather to take into account only the progressions. The result in this case seems better only because the progression of the piece incorporates precisely those cadences that were previously preferred through preprocessing operations.

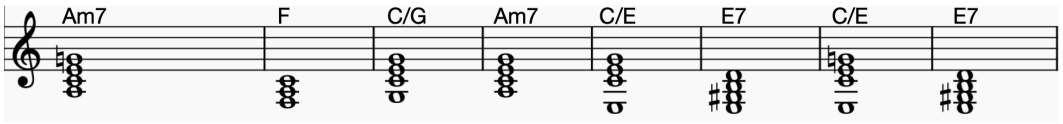

Figure 5.16: Chords progression chosen by the model for the second stanza

Set 2

The second set of parameters, as seen previously, is essentially the opposite of the first. The imbalance in weight leans towards the *bonus_note*. Concerning the first stanza, the model produces the following chord sequence:

- A Minor 7 - D Minor - C/G - C/E - Am7 - E Major 7 - C/E - E Major 7.

  In the figure 5.17, the resulting score for the first 8 chords is shown, regarding this parameter setting.

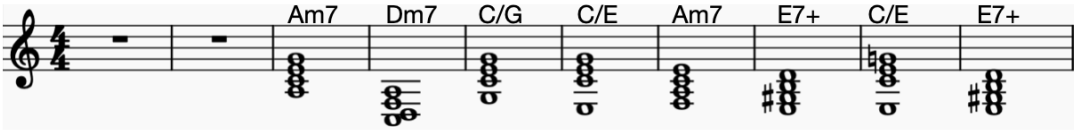

Figure 5.17: Chord progression chosen by the model for the second stanza

We can observe from the musical sheet that, similar to the first set of parameters, the first and last chords of the progression are correctly predicted by the model. However, in this case, the model makes another correct prediction:

- The second chord of the original progression, namely D minor 7, is also generated by the model.

This difference from Set 1 can be attributed to the model's increased ability to follow the melody as the *bonus_note* parameter grows. Observing the figure, it can be noted that with the aforementioned parameters, the model, in the central part of the progression, similar to the other tests presented, generates repetitions of chords one after the other. In particular:

- The third and fourth chords are two inversions of C major, namely C/G and C/E.
- The fifth and sixth are two E major chords respectively with and without the seventh.

When the parameter concerning the melody has too high a value compared to the parameter concerning the cadences, the model generates chords solely based on the melody. As the melody in this case is repetitive, with some variations excluded, so are the chords. The model's behavior under these conditions is confirmed by the second stanza. This can be observed on the sheet music depicted in the figure 5.18 and resents the following progression:

- A Minor 7 - D Minor 7 - C/G - A Minor 7 - C/E - E Major 7 - C/E - E Major 7.

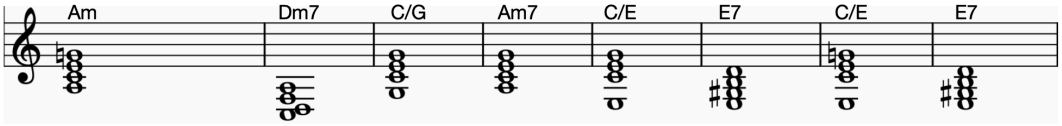

Figure 5.18: Chord progression chosen by the model for the first stanza

Here too, the first, second, and last chords coincide with the original progression, but the other chords are incorrect, and once again, the inversion chords of C major are repeated. It can be concluded that using parameter set number 2 on this piece results in similar issues, though less pronounced.

Set 3

The third set of parameters in previous tests had improved the model's behavior. In this piece, however, the model performs worse compared to sets 1 and 2. Specifically, for the first stanza, the following chord progression is obtained:

- A minor - D minor 7 - C/G - C/E - F major 7 - E minor 7 - C/E - E major 7.

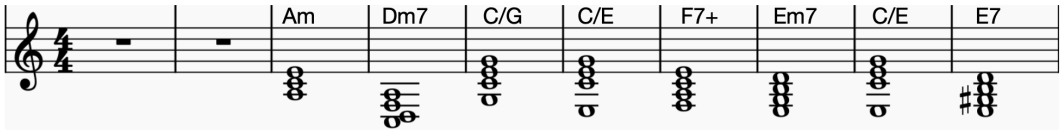

Figure 5.19: Chord progression chosen by the model for the first stanza - 3rd set

Comparing the score above 5.19, which is generated by the model for the first stanza, with that of the original progression 5.13, we can notice that the first and second chords, namely A minor and D minor 7, are correctly generated. The same applies to the final E major 7 chord of the progression. However, it should be emphasized that here too the model fails to predict the authentic cadence between

the third and fourth chords, which in the original is given by the succession of G7 and C major 7 chords. Furthermore, the model not only fails to predict the original chords but also includes two inversions of C major which we intended to penalize. Additionally, there is another repetition, using two chords with E as the root one after the other.

- A Minor 7 - F Major - C/G - A Minor 7 - C/E - E Major 7 - C/E - E Major 7

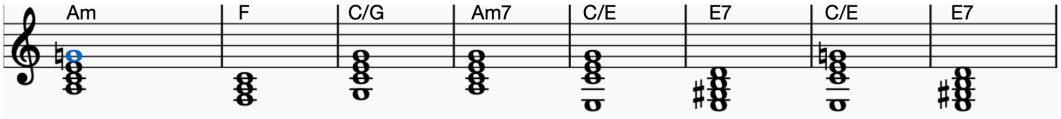

Figure 5.20: Chord progression chosen by the model for the second stanza - 3rd set

Observing the figure 5.20, we note that similarly to the first stanza, the model correctly predicts the first and last chords of the original arrangement, namely A minor and the final E major seventh.

In contrast to the first stanza, here the second chord, which should be a D minor 7, is instead an F major. As explained previously, these two chords share 3 common notes, and therefore, although F is not the chord in the original accompaniment, we can consider it a good substitute for D minor. The same concept applies to the fourth chord; the C major of the original progression is replaced by an A minor 7. Upon closer analysis of this chord, we see that A minor 7 is formed by the following notes:

- A - C - E - G

The last three notes are exactly the same as those forming the C major chord, and for this reason, we can also consider this substitution as partially accurate.

As for the subsequent chords, they are considered incorrect in terms of comparison with the original piece since those produced by the model and those of the progression are too different.

Set 4

This set of parameters, in previous pieces, consistently yielded better results compared to other parameter combinations and so it happens in this case . In this case, the similarity to the original chord progression improve with these parameters. Analyzing the first and second stanzas more closely, the results are as follows:

- - The first stanza consists of the chords A minor 7 - F major - C/G - C/E - F major 7 - E minor 7 - C/E - E major 7. This can be observed on the music sheet below 5.21s

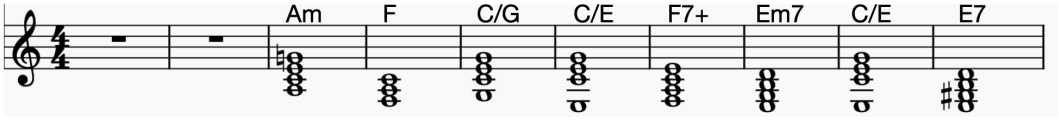

Figure 5.21: Chord progression chosen by the model for the first stanza - 4th set

The first and last chords, as in all other cases, are correctly generated by the model. Furthermore, as happened previously, the D minor 7 is partially correctly substituted with the F major chord. However, the model makes two additional predictions here that increase the similarity with the original accompaniment:

- The third chord is an inversion of the C major chord with G in the bass; observing figure 5.13, we notice that in the original, there is a G dominant seventh. Although the chord is not the same, the model remains somewhat faithful to the bass line.

- As for the fourth chord of the progression, in the original piece, it is a C major 7, and consequently, the model's choice of C/E can be considered correct.

- The fifth chord of the progression is also correctly generated by the model. In fact, both in the original and in the produced score, it is an F major 7.

The first stanza is therefore predicted almost entirely correctly by the model. In the second stanza, however, the model slightly worsens, failing to accurately predict the following aspects:

- The fourth chord reverts to being an A minor 7 instead of a C major 7, replicating the error in progression generation that also occurs in set number 3.

- The F major 7 chord, which should be the fifth chord of the progression, is here replaced by a very different chord, C/E.

To summarize, we can state that, regarding the song "I Will Survive," the results obtained by testing the various parameter sets are less disparate compared to other sets. This is due to the number of melody notes relative to other pieces.

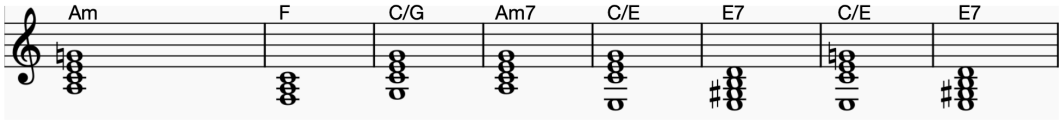

Figure 5.22: Chord progression chosen by the model for the second stanza - 4th set

5.5 Result analysis and discussion

In this section, we will analyze and summarize the results obtained from the previous tests. Firstly, it is worth noting that the choice of parameters plays a crucial role in determining the chords generated by the model. Therefore, it can be said that we are able to modify the behavior of the model and steer it towards certain choices based on the parameters. Let's summarize the general behavior of the model based on the four sets used:

- The results obtained with the first set of parameters, where the *bonus_progression* is set to 15 and the *bonus_note* is set to 5, exhibit a notable imbalance between the consideration given to chord progressions versus melody notes. This imbalance is reflected in the generated chord progressions, which prioritize common progressions while largely disregarding the melody of the piece.

  The model tends to favor certain chord progressions, such as the E7 - Am (V-I) cadence, and frequently incorporates transitions like Am to F Major (Am - F), which are typical in popular music. However, this preference for common progressions comes at the expense of coherence with the original melody.

  Despite correctly selecting some initial and final chords, the model's choices often diverge from the original piece's harmonization, opting for chord substitutions that may sound acceptable but lack coherence with the melody. For instance, in some instances, the model replaces chords that are structurally similar to those in the original progression, emphasizing the importance of harmonic consistency but neglecting the melodic context.

  Overall, the results underscore the need for a balanced consideration of both chord progressions and melody notes in generating harmonization. In this particular case, the disproportionate weight assigned to chord progressions leads to a disconnect between the generated accompaniment and the melody, resulting in harmonization that deviate significantly from the original piece

- The second set of parameters, where the *bonus_progression* is set to 5 and the *bonus_note* is set to 15, has imbalance towards the bonus related to melody notes compared to the first set. Consequently, the model tends to

select chords that closely follow the melody, often resulting in sequences of almost identical chords in succession.

This tendency is observed in the generated chord progressions, where the model consistently chooses chords that mirror the melody, sometimes at the expense of adhering to typical chord progressions or cadences. For example, the model may opt for chord substitutions that align with the melody but deviate from the original harmonic structure of the piece.

Despite correctly predicting some chords that match the melody, such as inversions of C major or chords that share common tones with the melody, the model's choices can deviate from the original progression, particularly when it comes to harmonic transitions and cadences.

Overall, the use of parameter set number 2 results in chord progressions that closely follow the melody but may lack coherence with the original harmonic structure of the piece. While this approach reduces repetition compared to the first set, it still presents challenges in accurately harmonizing the melody while maintaining harmonic consistency.

- In general, the results obtained with the third set of parameters, where both $bonus_progression$ and $bonus_note$ are set to the same value, indicate a balance between the weights. This balance was expected to lead to a more uniform distribution between the consideration given to chord progressions and melodic notes compared to previous sets. However, comparing the chords generated by the model with the original ones, several observations can be made.

  Firstly, the model's chord progression begins to resemble the original progression, but the result varies depending on the piece being analyzed. The explanation for this behavior can be found in the fact that each piece has a certain number of melody notes for each beat. The preference for certain chords assigned to beats based on the melody, as explained earlier, is guided by the weight assigned to the edges of the graph leading to the chord.

  These weights are increased or decreased based on $bonus_note$, multiplied by the number of common notes between the chord and the melody. This means that for pieces with many melody notes per beat, the parameter value must be kept lower to achieve the same degree of chord penalty or preference. In practice, to keep the multiplication value $bonus\_note*common\_note$ constant, if the latter increases based on the average number of notes within a beat, then the former must be lowered.

  Despite this set enabling the model to more faithfully follow the original accompaniments, at some points, the model's choices follow the melody too closely, resulting in consecutive inversions of the same chord, which was intended to be avoided due to its repetitive nature. Comparing the performance

of the third set with those of the previous ones, it is evident that while the balance of weights improves the model's ability to generate chords more similar to the original ones, it still lacks in accurately grasping some harmonic aspects and avoiding repetitive patterns.

- The fourth set of parameters proves to be the best in balancing the factor related to the melody and that concerning chord progressions. As observed in the previous parameter values, in this case too, the model correctly substitutes the first and third chords of the original progression with chords very similar to the originals.

  However, variations in the model's progression compared to the original song's sequence introduce some additional chords. Although these variations enrich the variety of harmonic colors within the song, they do not exactly reproduce the cadences of the original composition. Furthermore, in this case, the model correctly avoids repetitions or sequences of similar chords. Compared to set number 3, indeed, the *bonus_progression* penalizing these aspects has been significantly increased. We can conclude by stating that this set of parameters effectively imitates the original accompaniments, predicting most of the chords.

Set	bonus_progression	bonus_note
1	15	5
2	5	15
3	5	5
4	10	5

Figure 5.23: Set of parameters for testing

The result obtained with the 4 combination of parameters are summarized in figures 5.26,5.24 and 5.27

Indeed, personalized combinations of parameters tailored to each individual piece are crucial for achieving the best results in automated music generation. While general trends and insights can be drawn from analyzing the impact of different parameter sets, the unique characteristics of each musical composition necessitate a more nuanced approach.

By experimenting with various combinations of parameters and refining them based on the specific requirements and nuances of the composition at hand, it is possible to optimize the model's performance and improve the coherence between

Set	Analysis of Parametric Effects - My Way
1	The generated chord pairs, such as E dominant 7 to A minor and F major to C major, lack coherence with the given melody. This is due to the strong imbalance in weights, where the parameter intended to adapt the accompaniment to the specific melody has little to no effect. Despite sounding acceptable in succession, the generated chords deviate from those used in the original piece, highlighting the importance of a more balanced weight distribution.
2	The model closely follows the input melody. The chord progression aligns with the melody notes, although there are deviations from the original chords. The proportion of parameter values works better here, resulting in a less repetitive sequence with varied chord choices. The model accurately predicts certain chords, such as the secondary dominant, but deviates from the original chords as the piece progresses, emphasizing the challenge of closely adhering to the melody.
3	The model generates chords that closely resemble those produced by set 2, demonstrating effectiveness in imitating the original progression. The balanced weights contribute to a more faithful recreation of the original chords, especially in capturing the secondary dominant and cadences.
4	This stands out as the set generating chords most similar to the original accompaniment. It successfully predicts chords for the first stanza, incorporating a secondary dominant while avoiding overly repetitive progressions. In the final part of the second stanza, set 4 accurately generates the sequence of chords, including G major, D minor, and C major. This parameter set demonstrates a better balance between melody and progression factors, leading to more accurate and faithful chord predictions.

Figure 5.24: Set of parameters for testing

the generated chord progressions and the original melody. Ultimately, the pursuit of the best result in automated music generation also requires the willingness to experiment with parameter combinations and iterate on the model's outputs until the desired outcome is achieved.

Certainly, the use of optimization techniques for music generation, through parameter tuning, has proven to enable not only accurate prediction of the original chords but also the ability to replace them coherently and creatively. This approach offers a significant advantage as it provides meaningful control over the creative

Set	Analysis of Parametric Effects - Someone Like You
1	The first set of parameters results in a model-generated chord progression that heavily emphasizes chord variations related to chord progression rather than melody. This imbalance leads to a lack of coherence with the analyzed piece, as the chord progression proceeds without considering the melody, creating a disconnect between the two elements.
2	In contrast to Set 1, Set 2 exhibits an imbalance towards the “bonus_note” parameter, resulting in a model-generated chord progression closely following the melody notes.The model tends to generate almost identical chords in succession, reflecting the pattern in the melody. Despite efforts to penalize certain progressions through increased weights, the higher weight assigned to “bonus_note” diminishes the effect of penalization. The model generates repetitive chords that align with the melody but may not adhere to the desired cadences and progression rules.
3	Set 3 features balanced weights of the parameters aiming for a more harmonious chord progression.The model's output starts to resemble the original chord progression of the analyzed piece, with correct choices for the first and third chords. However, challenges persist as the model, at times, closely follows the melody and fails to avoid penalized chord transitions, leading to repetitive and monotonous sequences.
4	The last set successfully replaces the first and third chords of the original progression with similar chords, demonstrating an understanding of harmonic structure. However, variations in the model's progression, introducing additional chords, may enrich harmonic colors but deviate from the original cadences.although the harmonization chosen by the model is more complex than the original one, the model selects chords which can still be replaced with those of the song.

Figure 5.25: Effects of Different Parameter Sets on the Song ’Someone Like You’

process, allowing users to shape the final outcome according to their personal tastes and the specific needs of the composition.

The ability to adapt the model’s parameters to the unique characteristics of each piece allows for a more accurate and sensitive interpretation of the music, thus contributing to achieving higher-quality results. This level of personalized control not only promotes fidelity to the original composition but also provides an opportunity to explore new interpretations and creative variations.

The use of optimization techniques and parameter tuning in automated music

Set	Analysis of Parametric Effects - Someone Like You
1	The first set of parameters results in a model-generated chord progression that heavily emphasizes chord variations related to chord progression rather than melody. This imbalance leads to a lack of coherence with the analyzed piece, as the chord progression proceeds without considering the melody, creating a disconnect between the two elements.
2	In contrast to Set 1, Set 2 exhibits an imbalance towards the “bonus_note” parameter, resulting in a model-generated chord progression closely following the melody notes.The model tends to generate almost identical chords in succession, reflecting the pattern in the melody. Despite efforts to penalize certain progressions through increased weights, the higher weight assigned to “bonus_note” diminishes the effect of penalization. The model generates repetitive chords that align with the melody but may not adhere to the desired cadences and progression rules.
3	Set 3 features balanced weights of the parameters aiming for a more harmonious chord progression.The model's output starts to resemble the original chord progression of the analyzed piece, with correct choices for the first and third chords. However, challenges persist as the model, at times, closely follows the melody and fails to avoid penalized chord transitions, leading to repetitive and monotonous sequences.
4	The last set successfully replaces the first and third chords of the original progression with similar chords, demonstrating an understanding of harmonic structure. However, variations in the model's progression, introducing additional chords, may enrich harmonic colors but deviate from the original cadences.although the harmonization chosen by the model is more complex than the original one, the model selects chords which can still be replaced with those of the song.

Figure 5.26: Effects of Different Parameter Sets on the Song ’My Way’

generation not only enables good results but also offers flexible and personalized control over the creative process, allowing users to fully express their artistic vision and achieve satisfying musical outcomes.

To further improve the model, we could consider extending its functionality to encompass a wider variety of rhythmic patterns, scales, and chords. Currently, some of these aspects have not been considered to keep the complexity at a manageable level. Introducing such elements could allow for a more specific customization of musical arrangements and enhance the model’s accuracy in predicting the original

Set	Analysis of Parametric Effects - I Will Survive
1	The model demonstrates a consistent ability to correctly identify the initial and concluding chords of the progression. However, it displays a notable inconsistency in its treatment of the D minor 7 chord, opting instead for F major. While it is acknowledged that these chords share common notes, the substitution lacks fidelity to the original progression. Additionally, a tendency towards chord repetition is observed, likely influenced by the prominence of the progression parameter in the model's decision-making process.
2	The model continues to exhibit accuracy in predicting the initial and final chords of the progression. However, it also encounters issues with the D minor 7 chord, opting for D minor instead. Despite sharing common notes, this substitution diverges from the original progression. Furthermore, there is a recurrence of chord repetition, possibly influenced by the heightened weight assigned to the melody parameter.
3	Consistent success is observed in predicting the commencement and conclusion chords of the progression. Nevertheless, the model inaccurately substitutes G7 with C/G and Cmaj7 with Fmaj7. Moreover, it fails to capture the authentic cadence between the third and fourth chords, affecting the fidelity of the generated progression. Additionally, there's a propensity for chord repetition, indicative of the prevailing influence of the progression parameter.
4	The model select in a correct way the first and last chords of the progressione. However, it still encounters a challenge with the D minor 7 chord, instead opting for D minor, though the chords share common notes. Notably, there is an improvement in fidelity as the model accurately predicts an inversion of the C major chord with G in the bass, aligning closely with the original progression. Despite this, there's a minor regression in the second stanza, where the model substitutes the expected F major 7 chord with C/E, deviating significantly from the intended progression.

Figure 5.27: Effects of Different Parameter Sets on the Song 'I Will Survive'

chords of a song.

Moreover, it would be interesting to involve users in the pre-processing stage, allowing them to indicate preferred progressions or musical genres. In our case, cadences between certain chords were all treated the same based on the *bonus_progression* parameter; however, further personalizing this aspect, such as assigning different weights for each progression, could more precisely guide the model to better fit individual user preferences.

Despite the current limitations, it is evident that the model, when properly parameterized, can structure the accompaniment of the song partially correctly.

This demonstrates both the utility of optimization techniques in this research field and the potential of this method to achieve highly personalized harmony.

Bibliography

[1] Nailson dos Santos Cunha, Anand Subramanian, and Dorien Herremans. «Generating guitar solos by integer programming». In: () (cit. on pp. 8, 28).

[2] *Music21 Documentation*. URL: https://web.mit.edu/music21/doc/ (cit. on pp. 8, 40).

[3] Dorien Herremans, Ching-Hua Chuan, and Elaine Chew. «A Functional Taxonomy of Music Generation Systems». In: (Year). Singapore University of Technology and Design, Agency for Science Technology and Research (A*STAR) & Queen Mary University of London; University of North Florida; Queen Mary University of London (cit. on p. 13).

[4] Filippo Carnovalini and Antonio Rodà. «Computational Creativity and Music Generation Systems: An Introduction to the State of the Art». In: (Year). Department of Information Engineering, CSC - Centro di Sonologia Computazionale, University of Padova, Padua, Italy (cit. on p. 13).

[5] Miguel Civit, Javier Civit-Masot, Francisco Cuadrado, and Maria J. Escalona. «A systematic review of artificial intelligence-based music generation: Scope, applications, and future trends». In: *Expert Systems with Applications* 209 (2022), p. 118190. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2022.118190. URL: https://www.sciencedirect.com/science/article/pii/S0957417422013537 (cit. on p. 13).

[6] Yueyue Zhu, Jared Baca, Banafsheh Rekabdar, and Reza Rawassizadeh. «A Survey of AI Music Generation Tools and Models». In: (2023). August 28, 2023 (cit. on pp. 13, 14).

[7] R. C. Pinkerton. «Information theory and melody». In: *Scientific American* 194.2 (1956), pp. 77–86 (cit. on p. 13).

[8] John A. Biles. «GenJam: A Genetic Algorithm for Generating Jazz Solos». In: *Information Technology Department, Rochester Institute of Technology* (Year). Associate Professor, jab@cs.rit.edu (cit. on p. 13).

[9] Manuel Marques, V Oliveira, S Vieira, and AC Rosa. «Music Composition Using Genetic Evolutionary Algorithms». In: *Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No. 00TH8512)*. Vol. 1. IEEE. 2000, pp. 714–719 (cit. on p. 14).

[10] Dorien Herremans and Elaine Chew. «MorpheuS: generating structured music with constrained patterns and tension». In: *IEEE Transactions on Affective Computing* (Anno) (cit. on p. 14).

[11] François Pachet. «The Continuator: Musical Interaction With Style». In: *Journal of New Music Research* 31.1 (2002), pp. 1–16. ISSN: 0929-8215 (cit. on p. 15).

[12] Anna Rita Addessi. «Esperimenti con una macchina musicale Il Continuator e i bambini di 3-5 anni». In: *Ricerche di Pedagogia e Didattica* 4.2 (2009). Didattica e Nuove Tecnologie (cit. on p. 15).

[13] Christine Payne. *Musenet*. Accessed: 2023-05-15. 2019 (cit. on p. 15).

[14] Seth* Forsgren and Hayk* Martiros. «Riffusion - Stable Diffusion for Real-Time Music Generation». In: (2022) (cit. on p. 16).

[15] Jean-Pierre Briot and François Pachet. «Music Generation by Deep Learning – Challenges and Directions». In: (Year) (cit. on p. 16).

[16] Kemal Ebcioglu. «An Expert System for Chorale Harmonization». In: *Department of Computer Science, 226 Bell Hall, State University of New York at Buffalo* (1987) (cit. on p. 17).

[17] Google AI. *Magenta*. Accessed 10 February 2023. n.d. (Cit. on p. 17).

[18] Florian Geiselhart, Frank Raiser, Jon Sneyers, and Thom Frühwirth. «MTSeq: Multi-touch-enabled CHR-based Music Generation and Manipulation». In: *Proceedings of the International Computer Music Conference*. Aug. 2010 (cit. on p. 17).

[19] Li-Chia Yang, Szu-Yu Chou, and Yi-Hsuan Yang. «MIDINET: A Convolutional Generative Adversarial Network for Symbolic-Domain Music Generation». In: () (cit. on p. 17).

[20] Gino Brunner, Yuyi Wang, Roger Wattenhofer, and Jonas Wiesendanger. «JamBot: Music Theory Aware Chord Based Generation of Polyphonic Music with LSTMs». In: (2023) (cit. on p. 18).

[21] Yi-Hsuan Huang Yu-Siang Yang. «Pop Music Transformer: Beat-based Modeling and Generation of Expressive Pop Piano Compositions». In: (2023) (cit. on p. 18).

[22] Massimo Varini. *Manuale di Armonia e Teoria* (cit. on pp. 20, 23, 24).

[23] Andrea Avena. *Teoria & Armonia* (cit. on pp. 21, 25).

[24] Michael Johnson. *Harmony, Composition and Arranging* (cit. on pp. 25, 27).

[25] *Berklee College of Music*. URL: https://cloud.info.berklee.edu/berklee-viewbook?campaign_id=7010Z000001ZkLHQA0&utm_source=google&utm_medium=cpc&utm_campaign=bcm-ug-google-ev-pmax-general-brand&gad_source=1&gclid=Cj0KCQiAqsitBhDlARIsAGMR1RjskkFG9OZibduFyYYMr-VguEwDg8ug_NtfGaTpNEnadyp5sEUYR0AaAmP0EALw_wcB (cit. on p. 35).

[26] *Python MIP Documentation*. URL: https://python-mip.readthedocs.io/en/latest/ (cit. on p. 39).

List of Figures