GiTab: Automatic Tablature Music Composition for Classical Guitar

Martim Zanatti dos Santos Gomes da Silva

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisors: Prof. Helena Sofia Andrade Nunes Pereira Pinto
Prof. David Manuel Martins de Matos

Examination Committee

Chairperson: Prof. Alberto Manuel Rodrigues da Silva
Supervisor: Prof. Helena Sofia Andrade Nunes Pereira Pinto
Member of the Committee: Prof. Fernando Amílcar Bandeira Cardoso

October 2021

Agradecimentos

Firstly, I would like to thank my two dissertation supervisors, Professor Helena Sofia Pinto and professor David Martins de Matos, for supporting, helping and guiding me in the right direction on the most important project of my life to date.

Second, I would like to thank two great friends of mine. To thank Filipe Azevedo for, first, making my time at IST a thousand times better, with his friendship and companionship. And second for helping me making this thesis possible. I had long conversations about the creation of the system with him, where he helped me put my thoughts in order. To Gonçalo Palma I want to thank for reading the thesis with me in order to correct some English errors. But mainly, thank him for the friendship we have, that brought me here.

I would like to thank my parents for all the support they have given me throughout all these years and for having insisted, during not so good times, that I not disconnect from my studies.

I would also like to thank Nicholas Ratcliffe, my guitar teacher, for teaching me everything I know about the instrument and about music. This was the only way I was able to pursue this thesis topic. I would also like to thank, Pedro Quintas, Danyal Valy, Francisco Braga and Francisco Gouveia for helping me evaluate the system.

Lastly, but not least, I want to extend my gratitude to all the colleagues and friends I made over the years, for helping me across different times in my life. A special thanks to my friends Fábio Faisca, Tomás Santos, Ricardo Chasqueira, Catarina Tareco, and Ana Nogueira.

Lisboa, October 11, 2021
Martim Zanatti dos Santos Gomes da Silva

I dedicate this work to all those
who give their lives for music and
guitar.

Resumo

Uma tablatura de guitarra consiste numa forma de notação musical que indica o traste e a corda sobre a qual cada nota deve ser tocada. Para guitarristas principiantes e guitarristas que não conseguem ler partituras, esta forma de notação é essencial. Na guitarra, ao contrário de outros instrumentos como o piano, cada nota existe em uma ou mais posições diferentes na guitarra. De modo que, quando se quer tocar um conjunto de notas, existem diversas maneiras de o fazer. O objectivo da tese é produzir tablaturas automáticas de música clássica, tentando optimizar as posições escolhidas para cada nota. Também é um objectivo desta dissertação escolher os dedos que pressionam os trastes de modo a produzir os sons das respectivas notas. O problema de gerar uma tablatura automaticamente, optimizando as escolhas das posições é um problema típico de IA. Dado que existem uma ou mais posições diferentes para cada nota, o crescimento das maneiras possíveis de tocar um conjunto de notas é exponencial. A nossa abordagem consiste na criação do sistema GuiTab que, através de regras e heuristicas, consegue optimizar a escolha das posições e excluir as hipoteses menos promissoras. Os resultados do nosso sistema foram comparados com tablaturas publicadas, onde obtivemos uma media de 68% de accuracy, valores parecidos com os obtidos por trabalhos baseados em Machine Learning. A avaliação subjectiva mostrou-nos que as tablaturas geradas pelo sistema são sempre tocáveis e, por vezes, são justificadas as escolhas do sistema que divergem das tablaturas publicadas.

Abstract

A guitar tablature is a form of musical notation that indicates the fret and string on which each note should be played. For beginner guitar players and guitarists who cannot read scores, this form of notation is essential. On the guitar, unlike other instruments such as the piano, each note exists in one or more different positions on the guitar. So, when we want to play a set of notes, there are several ways to do it. The goal of the thesis is to produce automatic tablatures of classical music by trying to optimize the positions chosen for each note. It is also a goal of this dissertation to choose the fingers that press the frets in order to produce the sounds of the respective notes. The problem of automatically generating a tablature by optimizing the choices of positions is a typical AI problem. Since there are one or more different positions for each note, the growth of possible ways to play a set of notes is exponential. We create the GuiTab system which, using a set of rules and heuristics, can optimize the choice of positions and exclude the least promising hypotheses. The results of our system were compared with published tablatures, where we obtained an average of 68% accuracy, values like those obtained by works based on Machine Learning. The subjective evaluation showed us that the tablatures generated by the system are always playable, and sometimes the system choices that diverge from published tablatures are justified.

Palavras Chave
Keywords

Palavras Chave

Nota
Hipóteses
Posição
Corda
Optimização
Caminho
Dedos
Guitarra
Música
Tablatura

Keywords

Note
Hypotheses
Position
String
Optimization
Path
Fingers
Guitar
Music
Tablature

Index

List of Figures

List of Tables

1 Introduction

Classical music originated after the fall of Rome around 500 a.D., remanescent of some influences dating from ancient Greece (Grout, 1960). The turning point that allowed us to conceive music as we do today was in the first period of classical music, the medieval period, where the introduction of the musical notation form allowed, for the first time, the separation of the musical composition from its transmission. Before, music was transmitted orally, which were subject to transmission errors. Since the medieval period, musical language has been under development until today. We can consider two types of musical notation: the standard musical notation, which corresponds to the sheet music and the musical notation known as tablature. The first gives us the notes' pitches while the second tells us which fret and string to press in order to get the desired note. Tablatures are easier to read than sheet music. However, their deficiencies, when compared to standard notation, made them lose their popularity in the years following their creation. Nowadays, tablatures are used mostly in musical genres such as rock, pop and blues. The difficulty in learning how to read a score, with all its musical nuances and the ease in learning to read a tablature, made this musical notation popular for the general public. With the easy access to video tutorials they gained further popularity, because tablatures provide the positions on the guitar where the notes should be played, and video tutorials help to understand rhythm and other musical nuances. In fact, there are several world-renowned guitarists that could not read scores, such as Paco de Lucia, Jimi Hendrix, Eric Clapton and Slash (Mistler, 2017).

Sheet music is mostly used in two musical genres: classical music and jazz, since to perform these two styles, professionally, requires musicians to have a degree in music or to have attended a conservatory. These two musical styles have a much higher degree of formality and detail than more popular styles like pop, rock and blues. Moreover, scores are more common in these two styles because of the theoretical musical detail present in scores as opposed to tablatures. Some might say that there are very few people who can fully read sheet music. This is even more uncommon in the aforementioned genres. Knowing how to read a sheet music requires a theoretical musical knowledge that is often not necessary in certain musical genres.

1.1 Motivation

There are several reasons why the results of this thesis are important and can add value. While for popular styles tablatures are easy to find, for other less popular genres like classical music, tablatures are hard to find and sometimes do not have enough quality (Tuohy, 2006). Creating a tablature manually is complex, very time consuming and requires great musical knowledge. As in our case they are tablatures for guitar, it also requires a great knowledge of the guitar instrument. Being able to automatically generate good quality tablatures would make it much easier to obtain them and would be very beneficial for learning novice guitar players. From

a more theoretical point of view, the main motivation is to be able to answer two questions that seem pertinent to us. Is it possible for the system to present results as good as humans? Knowing that there are countless ways of playing the same notes is it possible to define the best set of positions or there are a set of tablatures that are the best?

Most tablatures do not indicate which fingers to use for each note. However, we also cover the choice of fingers, because defining which fingers to use also defines whether a hypothesis of notes is more appropriate than another. In other words, it allows us to have a greater understanding of which hypotheses to choose from. Setting the fingers for each note is also educationally advantageous, as it allows novice students to not create bad habits of playing, which is quite common when learning an instrument on their own.

Another motivation is due to the complexity of the problem and the fact that it is still an unsolved one. There are already commercial products such as *Frettable* (Frettable, 2018), however, they are unreliable. Regarding the second reason, we think we are taking important steps in the right direction, that is, create reliable tablatures. As for the first reason, we can easily understand the complexity of the problem we have. Starting from the premise that each note on the guitar can be played in several positions, between one and five different positions, let us think about how many ways are possible to play a single music with a total of, for example, thirteen notes.

To understand the dimension of the problem, consider that we start with two notes, each one with four different positions to play in the guitar. We have 4^2 different ways to play these two notes. If the third note also has four different positions, we have to add the four different positions of the third note for each path of the sixteen previously calculated, so we have 4^3. If we consider, as an example, that in a music there are only notes with four different positions, we get that the growth of the different ways of playing the music is given by the following function.

$$y = 4^x = 2^{2x} \tag{1.1}$$

This function is shown in the figure 1.1. The complexity of this function is $O(2^n)$ and has an exponential growth. In a music where there are only thirteen notes where each one has four different positions, the number of possible ways of playing is sixty-seven million one hundred and eight hundred and sixty-four times. This is a typical complex AI problem: it is impossible to process all possible paths systematically. So, we need to make choices and exclude less promising paths. These choices emphasize the complexity of the problem and are discussed later in detail.

1.2 Goals

The main objective of the work reported in this thesis is to produce a tablature receiving as input a single guitar MIDI file. Unlike the piano, on the guitar, notes can be played in more than one position. With this in mind, and knowing that the tablature gives us the positions for each note, the aim of this work is to find the most optimal way to play the desired notes while also assign a specific finger out of the four possible ones.

The system receives as input a MIDI audio file, with only classical guitar and returns as output a tablature of the respective MIDI file. Therefore, the evaluation of other instruments

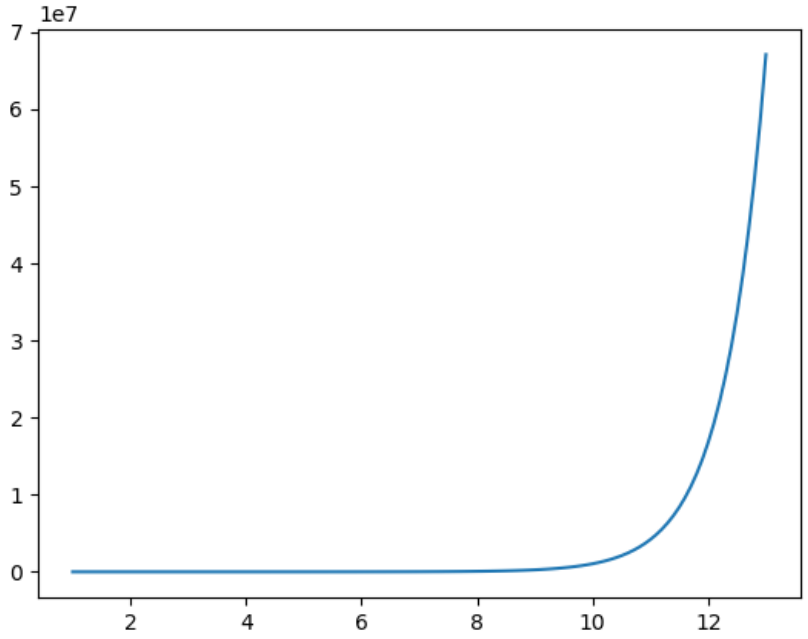

Figure 1.1: Function that describes the growth of possible paths to play a set of notes.

than classical guitar are not included in this thesis and all notes in the music are played by a single instrument.

MIDI audio files used to illustrate our work, are predominantly classical guitar music. First, because it is a genre with a vast number of musical pieces. Finally, because in this style one guitar plays like an orchestra, where the melody, accompaniment and bass are played simultaneously.

1.3 The Importance of Tablatures Today

Tablatures are important since they help in the development of novice guitarists. As you can see in the picture 1.2, a tablature consists very simply of a simplified abstraction of a score that allows beginners, who still find it difficult to decide in which positions to play each note, a conscious and efficient way of learning. This happens because tablatures give an optimized way to play a music. Besides being very useful for beginners, it is also very helpful for more advanced guitarists, who already have a very good knowledge, to use them as a facilitating and auxiliary tool.

Tablatures are increasingly common in more popular styles, and scores are rarely spoken when we learn music in these genres. It must be mentioned, however, that having only a tablature as a learning tool is not possible to play any music. This is because tablatures simply give us the notes and their positions and say nothing about the rhythmic value of each note nor the time of the music in question.

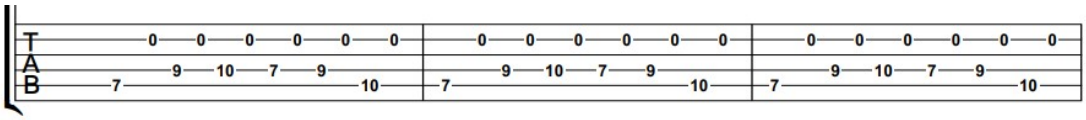

Figure 1.2: Beginning of Asturias by Isaac Albéniz

1.4 Structure of the Document

The remainder of this document is organized as follows: In Section 2 we introduce the Background Concepts, which explain the musical concepts used throughout this paper. Section 3 presents relevant related work from the field of automatic composition. In section 4 we describe how we implement our system. In Section 5 we present the evaluation metrics, our dataset and the results of our work. Finally, Section 6 offers some conclusions and what can be done in the future to complement the work that was done and presented here.

2 Background Concepts

In this section we present the background concepts needed for a better understanding of our problem and its domain. This chapter is divided in three sections: Musical concepts, section 2.1 where we introduce musical theory focused for classical guitar; Guitar concepts, section 2.2 that explains guitar concepts essential to understand the solution of our problem and finally in section 2.3 we explain MIDI audio files.

2.1 Music Concepts

Since the goal of our thesis is the automatic production of tablatures from a MIDI guitar audio file, it is essential to understand musical concepts, both those common to several instruments, as well as concepts that are specific for guitar. Without these basic concepts it would be difficult understand the dimension of the problem and its several possible approaches. First, we define general concepts of music. Then we specify the important characteristics in a sheet music and the important characteristics in a tablature. We conclude by comparing the differences between a tablature and a sheet music.

2.1.1 Music Theory

2.1.1.1 Pitch

Pitch is the fundamental frequency of a note. The smallest increment of a pitch in Western music is a half-step. In figure 2.1 there is a representation of a keyboard. From a white note to the next black note the increment is a half-step. From a white note to the next white note the increment is a step. In figure 2.2 we can see a C octave. An octave is the largest repeating of pitch, twelve half-tones, so seven natural notes and five accidental notes. A pitch follows a logarithmic scale: one octave above the other has double the frequency. For example, A4, which is the fourth octave of A has a frequency of 440 Hz, which means that A5 has a frequency of 880 Hz and A3 has a frequency of 220 Hz.

There are several common guitar techniques that use the variation of frequency to add expression to the music and that are not perceptible in MIDI audio files, but that influence the positions of the notes to be played: Vibrato, Tremolo and String bending. Vibrato, a common technique usual in guitar playing, is a regular pulsating change of pitch.

Tremolo, which is the rhythmic change in loudness, is created by playing a bass note with the thumb (p) followed by three repeated higher notes, traditionally the ring (a), middle (m),

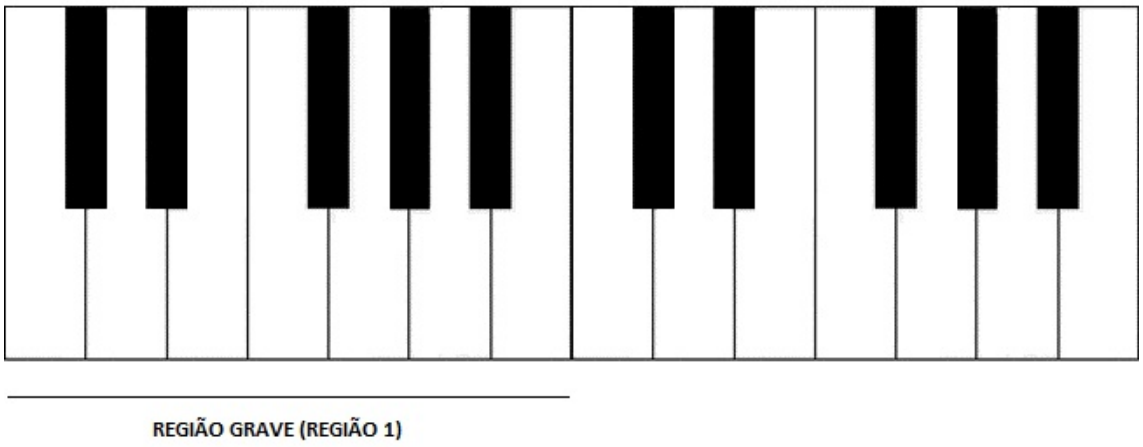

Figure 2.1: Representation of a keyboard.

and index (i) fingers play the higher notes. When played quickly, this technique creates the illusion of a sustained upper line with a bass accompaniment, which implies that each note of these three must be played in the same position. Figure 2.3, shows an excerpt from one of the most famous pieces of the classical guitar repertoire, *Recuerdos de la Alhambra* by composer *Francisco Tárrega* (Recuerdos Alhambra, 2019). It is also one of the most famous pieces using the tremolo technique. In this excerpt, the first note is the bass note, and the (p) above indicates that this note is played with the thumb. After this note, the same note is repeated three times, where the first is played with the ring finger (a), the second is played with the middle finger (m) and the last one with the index finger (i). After these three notes another bass note is played and so on.

Finally, string bending, more usual in electric guitar, consists of increasing a pitch of a note or notes which can result in two different notes due to the impossibility of the MIDI files to detect this technique.

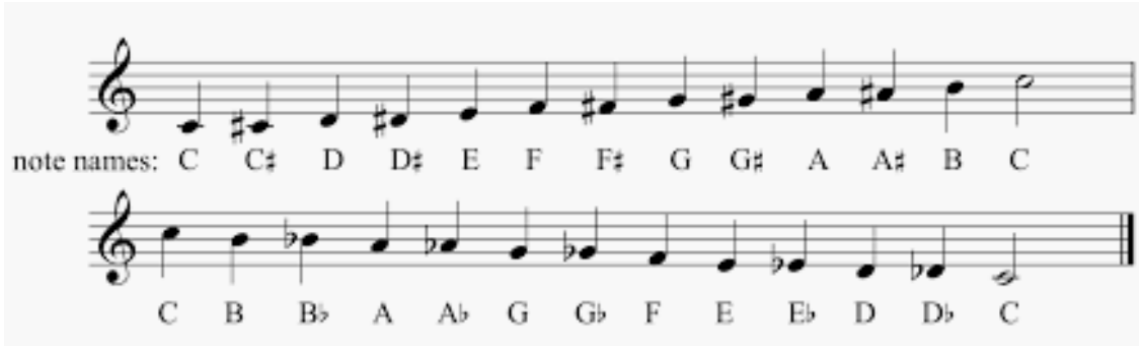

Figure 2.2: C octave.

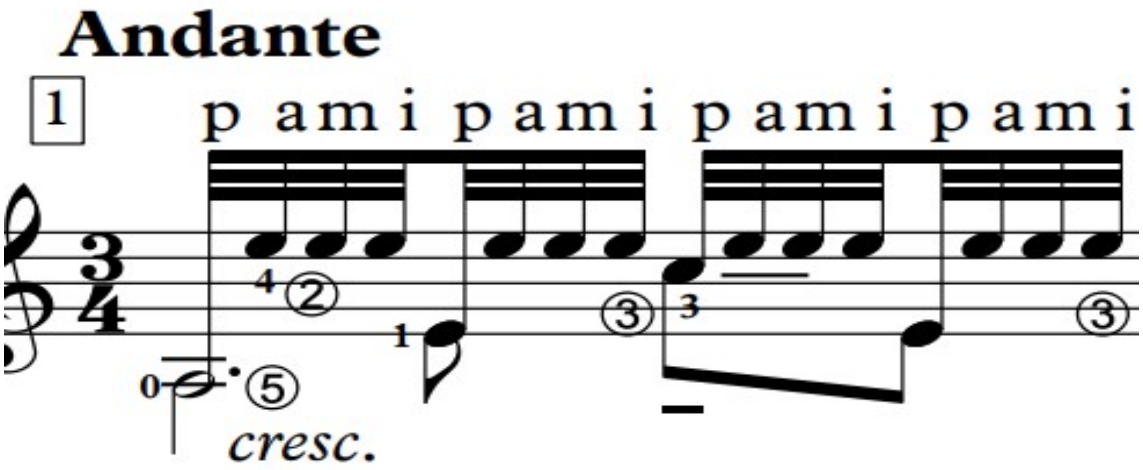

Figure 2.3: Recuerdos de la Alhambra by Francisco Tárrega.

2.1.1.2 Rhythm Figures

Rhythm figures are indispensable when the subject is music. They represent the duration of a certain note. Six rhythmic figures are represented in figure 2.4: semibreve, minim, crotchet, quaver, semiquaver and demisemiquaver. In figure 2.5 are represented the correspondent rests: semibreve rest, minim rest, crotchet rest, quaver rest, semiquaver rest and demisemiquaver rest. The semibreve has a duration of one, the minim has half the duration of a semibreve, the crotchet has a quarter of the duration of a semibreve and so on. There are more rhythm figures than these six, but these are the more usual ones for guitar and playing smaller notes introduce insurmountable problems.

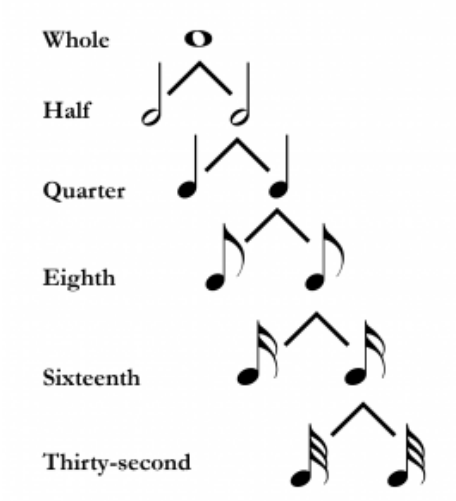

Figure 2.4: Rhythm Figures

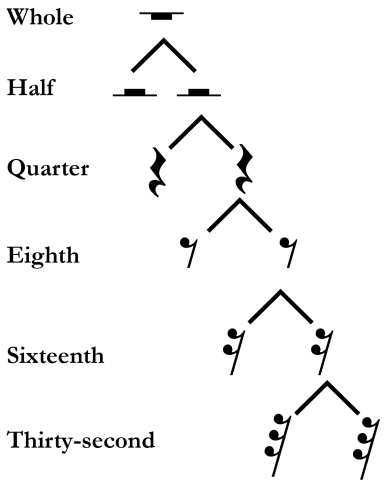

Figure 2.5: Rest figures.

2.1.1.3 Form

In music, form consists of several patterns that constitute a piece of music. These could be some repetitive elements that occur along the music such as a motif, which is defined by (Scholes, 1938) as "A melodic or rhythmic musical unit which reappears throughout a composition, either in its original form or at different pitches and perhaps with altered intervals". *Lágrima*, by *Francisco Tárrega* (Lagrima, 2018), is divided in two sections, that are repeated throughout the music, section A and section B. This Composition has a consistent structure, what is called a ternary form, ABA, with repeated sections, which does an AABBA structure. There are others structures, as binary form and rondo.

In the context of our problem, if we think that certain sections of the piece are repeated, mainly in pieces with a type of structure defined as the AABBA structure, it would be expected that each section A would be played in the same way and each section B would be equal. However, in our solution we do not identify or divide the piece into parts. Yet, we consider the repetition of notes, but in a shorter interval of the piece.

2.1.1.4 Rhythm

Pulsation, beat and time are the base of rhythm. A pulse is the heartbeat of the rhythm/music that you hear - and feel - when listening to music and this is what people usually tap along to when listening. The beat is the repeated note value of the time signature. They can often (and are usually) the same thing, or at least they cross over. For example, In a piece with time sig 4/4, the beat is 4 crotchet beats every bar. The pulse is most likely also going to be this however if some notes are more pronounced you may tap your foot 2 beats/bar. Time is the interval between two beats.

Tempo, which means time in English, refers to the speed of a given piece, and is measured in beats per minute (bpm). This *tempo* is usually specified in the beginning of the sheet music, and sometimes, could be elsewhere in it, depending on the circumstances of the piece. There are countless different types of *tempo* like adagio (55-65 bpm) or presto (180-200 bpm).

2.1.1.5 Dynamics

Dynamics consist of variations that occur in a piece. We can define three types of dynamics. Variations in loudness, also called Intensity Dynamics; variations on *tempo*, referred as Speed Dynamics; and variations on a single note, designated as Stress Variations. In figure 2.6, we can see these three types and some of its constituents. These indications are fundamental when performing a piece because they give information so that the interpretation reflects the atmosphere and emotion of a certain piece.

Due to the subjectivity of these dynamic, each performer has a different interpretation of the same piece. In classical music it is usual for the composer to compose his own dynamics. So, when the performer performs a certain piece, he/she must follow the dynamics proposed by the composer. Still, the result is the set of dynamics of the composer and the performer. In other genres, such as jazz, blues and rock, the performer normally has more freedom to perform as he wishes.

Intensity Dynamics		
Symbol	**Name**	**Meaning**
ppp	Piano-pianissimo	as soft as possible
pp	Pianissimo	Very soft
p	Piano	Soft
mp	Mezzo piano	Medium quiet
mf	Mezzo forte	Medium loud
f	Forte	Loud
ff	Fortissimo	Very loud
fff	Forte-fortissimo	Loud as possible
<	Crescendo	A gradual increase of intensity
>	Diminuendo	A gradual decrease of intensity
sfz	Sforzando	Indicates a forceful accent in a specific note or a set of notes
Speed Dynamics		
Symbol	**Name**	**Meaning**
Accel.	Accelerando	Gradually increase the tempo of a song until otherwise noted
Rit.	Ritardando	Gradually decrease the tempo of a song until otherwise noted
Stress		
Symbol	**Name**	**Meaning**
–	Tenuto	A note has to be played at full length
.	Staccato	A note of shortened duration
^	Marcato	shorter as Staccato louder as Sforzando

Figure 2.6: Dynamics

2.1.1.6 Timbre

Timbre can be defined as the harmonic content of a note, the color of a note, and the quality of a note. Each instrument has its own timbre. Guitar timbre is different from piano timbre, flute timbre, and so on. In other words, given a pitch played on a guitar, timbre accounts for the difference between this pitch and the same pitch played on a piano. Although timbre is affected mostly by the instrument choice, on the guitar could also be changed with mutes. To mute a string, we use the fretting hand, touching the string with one finger, but do not press it down, and strike the string. The idea is not to completely mute the strings, but to dampen them, so that the notes are still clear, but with less sustain. Another way to change timbre is where the right hand is placed for right-hand guitars. If the right hand is placed near the sound hole, the timbre is sweeter and if it is placed far from the sound hole, the sound is more metallic. Timbre is usually discussed in terms of richness and softness, where in a rich timbre the range of harmonics is bigger than a soft timbre, where higher harmonics are less present.

2.1.2 Sheet Music

In figure 2.7, we have the first three bars of the *Adelita* guitar piece composed by *Francisco Tárrega* (Adelita, 2017). As it can be seen, some concepts explained above are directly specified in this sheet music, like dynamics, the *tempo* of the piece and the rhythm figures.

A **Bar** is a way of organizing the writing music in sections, where each bar represents an amount of time. *Adelita* is a $3/4$ time signature, which means that for each bar, there are three crotchet or equivalent rhythm figures as explain in section 2.1.1. In other words, the numerator represents the quantity of a certain rhythm figure and the denominator represents the quality of the rhythm figure.

Clef, which is the first symbol present in figure 2.7, is an important concept, because it indicates the position of a pitch of a written note as reference. In this piece we have a G-clef, where the curl of the clef is placed on the second stave, which indicates that a G4 is positioned on that stave. Once we know the position of the G4 note, we know the positions of the other notes. The note following the G4 note is the A4 note where it is placed on the line just after the second stave and so on. In modern music notation the most common clefs are the G-clef, C-clef and the F-clef. In the case of piano scores, the score intended for the right hand is usually on the G clef while the left hand is usually on the F clef, this is because the G clef allows us to go through higher notes while the F clef allows us to go through lower notes.

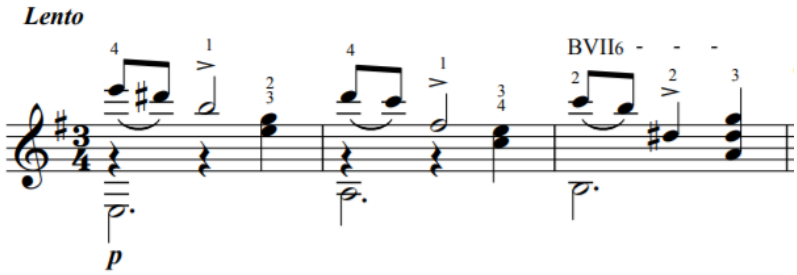

Figure 2.7: Beginning of Adelita by Francisco Tárrega

The *Lento* indication, that appears in the sheet music, gives us information about how fast we should play the piece. It is the *Tempo* of the music, as explained in the section 2.1.1.4. The # that appears near the G-clef indicates that the F notes are accidental. The >symbol indicates that the note underneath must be heard more than the rest of the notes. The remaining relevant indications are explained in the following sections.

2.1.3 Tablature for Guitar

Since the output of our program consists of a tablature for guitar it is essential to present the notations and characteristics inherent to tablatures. Tablatures are a very simplified abstraction of a sheet music. As it is possible to see in the figure 1.2, the first three bars of the *Asturias* guitar

piece composed by *Isaac Albéniz* (Asturias, 2019), the division is also made by **bars**, but instead of having rhythmic figures represented in each **bar** it has six horizontal lines, which correspond to each guitar string, from the highest to the lowest, and each note is presented with a number that represents the fret in which the note is played. Looking at the first note of the tablature, which we represent as (5.7), it tells us that this note is played on the fifth string and the seventh fret. As stated earlier, the tablatures do not contain information about the fingers of the left hand. However, we decided to add this information to the tablatures generated by the system.

2.1.4 Sheet Music for Guitar

In figure 2.7 there are some notations that are specific for guitar. The numbers near the notes indicate the finger which is supposed to use to play that specific note. The number one corresponds to the index finger, the number two correspond to the middle finger, the number three corresponds to the ring finger and the number four corresponds to the little finger. When a circle with a number inside appears next to a note or a set of notes, it indicates that these notes should be played on the string specified by the number. The BVII6 (or CVII6) indicates that in this position we must perform a barre. A barre consists of pressing one single fret, usually with the first finger, and use the remaining three fingers to play other notes. B stands for barre (C stands for cejilla which means barre in English), VII stands for the fret position and the number 6 stands for the number of strings that have to be pressed with the barre. These notations which are common in guitar sheets, help reading the music sheet and optimize the use of the left-hand for right-hand guitars. Given the fact that there are several ways to play the same note or conjunction of notes, some ways are more appropriate in a certain moment than others. In classical guitar, the unfretted notes facilitate a lot the technique of the left-hand, because for the unfretted notes it is not necessary to press any fret to play them, which we represent as $(s, 0)$, where the s correspond to one of the six strings.

2.1.5 The Differences Between a Tablature and a Sheet Music

As we mentioned before, tablatures are a simplified version of sheet music and this can be verified from figures 2.7 and 1.2. While the sheet music of *Adelita* contains a large set of information of musical theory, indispensable to play a piece in a correct way, the tablature presents us the same information focused for the guitar. In the tablature we do not have any information regarding the time of the music nor the rhythmic values of each note. We also do not have any information about dynamics as we presented in section 2.1.1.5. It has already been mentioned that it is not possible to play a piece correctly having only a tablature as a learning tool. However, this limitation does not take away the popularity and usefulness of tablatures, because together with other tools, tablatures become quite useful for the beginners or naive players.

2.2 Guitar Concepts

Since the goal of our work only address classical guitar MIDI files, and we aim at optimizing the way a set of notes is played on the guitar and which fingers should be used for each note, it is important to approach some concepts and specificities of the guitar, the instrument of our problem.

2.2.1 Guitar Characteristics

From here on we always assume that the guitar is built for right-handed guitarists, which means that the right hand plays the strings while the left hand presses the frets. As you can see in figure 2.8, the right hand is positioned in the body area while the left hand walks through the fretboard.

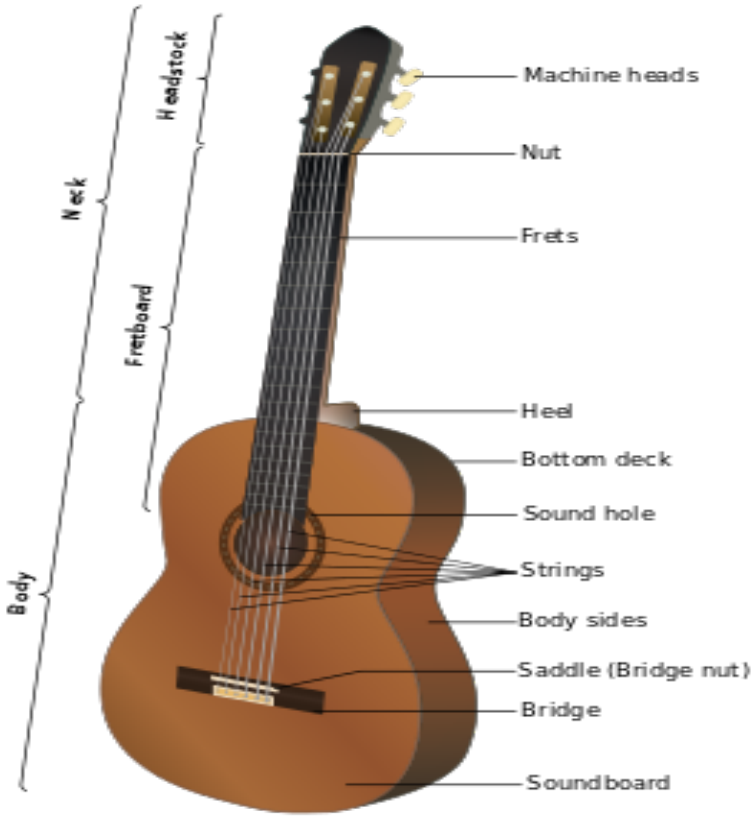

Figure 2.8: Guitar Characteristics

Figure 2.8 presents the physical characteristics of the guitar, considering that we are interested only in the fretboard that is where pressing a string in a certain fret makes a certain note sound. In this thesis we do not address the problem of the right hand, that is, we do not associate each finger of the right hand with which string to play, since the choice of fingers of the right hand also has its own set of rules and nuances.

2.2.1.1 Fretboard

We are mainly interested in fretboard given that the fretboard is where we press the frets to play the notes. Figure 2.9 represents an abstraction of a fretboard with the standard tuning. Each vertical line represents the division of the frets and from one horizontal line to the other, the notes of the respective strings are represented, the first is the highest and the bottom one is the lowest. For example, the note E4 is higher pitched than the note E2. The first fret represents the "fret 0" or loose note, that is, the note that is played when playing the string without pressing

any fret. So, the note E2 is on the sixth string, "fret 0" while the note F2 is on the sixth string, fret 1.

We work with fretboards with nineteen frets, which is the most common number in classical guitars. As it is possible to observe there are two notes with different colors, where we can observe the maximum and minimum of repeated positions for each note. The E2 note painted in green has only one position while the E4 note painted in red has five possible positions on the guitar.

For each note it is possible to play it with four different fingers, and we have decided to use the same notation explained in section 2.1.4. For the loose notes the finger notation will be the 0 finger, which implies that it is not necessary to use any finger of the left hand to play the respective note. The number one corresponds to the index finger, the number two correspond to the middle finger, the number three corresponds to the ring finger and the number four corresponds to the little finger.

In section 1.1 we looked at the complexity of the various ways of playing a set of notes. Now we look at the complexity of building the possible finger paths. We know that there are 4 possible fingers to play each note. So, when we have two notes it is possible to play it (in terms of fingers) in 16 possible ways: $(1,1)$, $(1,2)$, $(1,3)$, $(1,4)$, $(2,1)$.... Imagining a number n of notes we have that the possible paths are 4^n, the same function as in figure 1.1. Therefore, both the complexity in the choice of notes and the complexity in the choice of fingers are exponential.

E4	F4	F#4	G4	G#4	A4	A#4	B4	C5	C#5	D5	D#5	E5	F5	F#5	G5	G#5	A5	A#5	B5
B3	C4	C#4	D4	D#4	E4	F4	F#4	G4	G#4	A4	A#4	B4	C5	C#5	D5	D#5	E5	F5	F#5
G3	G#3	A3	A#3	B3	C4	C#4	D4	D#4	E4	F4	F#4	G4	G#4	A4	A#4	B4	C5	C#5	D5
D3	D#3	E3	F3	F#3	G3	G#3	A3	A#3	B3	C4	C#4	D4	D#4	E4	F4	F#4	G4	G#4	A4
A2	A#2	B2	C3	C#3	D3	D#3	E3	F3	F#3	G3	G#3	A3	A#3	B3	C4	C#4	D4	D#4	E4
E2	F2	F#2	G2	G#2	A2	A#2	B2	C3	C#3	D3	D#3	E3	F3	F#3	G3	G#3	A3	A#3	B3

Figure 2.9: Guitar Notes

2.2.1.2 Guitar Techniques

Besides the techniques aforementioned in section 2.1.1 there are three other techniques that are important to mention. The slide technique consists in sliding from one note to another on the same string either to a lower note or to a higher one while pressing the string along the slide. A pull-off is an articulation technique that consists of plucking or "pulling" the finger that is grasping the sound, allowing a new sound to be heard from the same string, that is from a lower note that is being pressed behind or the loose note of that respective string. Finally, a hammer-on which is the opposite of a pull-off, you are pressing a lower note or playing the loose note and you press a new higher note of the same string. These techniques work on the same string, so they are very useful in guiding the choice of the positions of the notes that are under these techniques.

2.3 Midi Files

A MIDI (Musical Instrument Digital Interface) is a technical standard that describes a communication protocol, digital interface, and electronic connectors that connect a wide variety of electronic musical instruments, computers, and related audio devices for playing, editing and recording music. To extract information from the MIDI file we used the python music21 library [1] which extracts the information present in the MIDI file. The MIDI file information extracted by music21 library is divided into three channels. Each channel has a set of notes. The first channel has the melody notes, the second the accompaniment notes, and the third the bass notes. Each note is an object of the Note class present in the music21 library. Each note object has a set of attributes such as the note's value, its duration, its onset. We explain this in more detail in the Method section 4.

The big difference between MIDI files and other types of audio files falls under the fact that the information present in MIDI files is discrete instead of continuous. So, we chose MIDI files as input because the information available in these files is much more manageable than other types of audio files as mp3 files. This is because with other types of audio files we would have to use multi-pitch estimation to get information from the notes and the problem of multi-pitch estimation is still an unsolved problem. It would be a problem of multi-pitch estimation because the songs that are evaluated by our system are not monophonic songs. That is, there is at some point in the music more than one note to be heard. Therefore, to better understand the robustness of our solution we decided to have as input something that allows us to have the necessary information to explore our problem.

[1] http://web.mit.edu/music21/

3 Related Work

3.1 Introduction

The task of producing a system capable of creating an optimized tablature of a guitar piece has been explored over the last decades but remains an unsolved and challenging problem. These systems are at the intersection of both artificial intelligence and the field of music theory. In the following sections we introduce some of the previous approaches that have contributed to progress in solving this challenging problem. We also introduce theory about the complexity of playing classical guitar that allowed us to create a robust set of rules that excels in optimizing note positions.

3.2 Complexity of Playing Classical Guitar

The movements that guitarists must perform in order to perform well a certain piece are unnatural and demanding movements that force awkward positions in relation to the biomechanical movements of our day to day. And, since western music is extremly strict in relation to time intervals, these positions have to be well and fast performed in a short period of time. In this paper (Heijink & Meulenbroek, 2002), the authors perform a behavioral study of the complexity of left hand movements when playing a classical guitar. For this, six professional guitarists were asked to play a sequence of notes at a fixed time. The authors examined the movements of the left hand in order to gain more insight into the biomechanical bases of the complexity of the movements led by the left hand in the sequence of requested notes.

When the positions of a set of notes are chosen to play on the guitar, that set propels a certain posture and a certain movement that have to be made by the left hand in order to perform that set of notes. There are three matrices that are governed when choosing the postures: the biomechanical movements needed to perform the task, the cognitive bases and the musical ones. The authors are only interested in the first matrix of the problem, and since they are trying to gain some insight into the biomechanical basis of the complexity of the movements led by the left hand, they also know that there is a parameter of subjectivity which they have to take into account, because different guitarists may have different difficulties in performing certain movements given the guitarists' own characteristics. In this paper, the authors consider that the guitarists choose the set of fingers that most facilitates the biomechanical process. This choice can be seen as the search for the lowest possible cost to accomplish the task.

Previous research in this supports the hypothesis that the postures chosen to perform a task are chosen based on the minimum-cost principle (Rosenbaum, 1995) and (Rosenbaum, 2001).

The authors in this paper (Cruse, 1990) tell us that the cost of a posture is lowest when the joints are in the middle of their ranges and increases nonlinearly when the joint angles leave the middle of their range. In this paper (Rosenbaum, 1996), the authors show empirical evidence that wrist oscillation around the axis of the forearm can be done faster when the forearm is in the middle of the pronation-supination range. Since playing guitar requires fast and precise movements of great oscillation it is possible that this is the reason why guitarists prefer to keep their joints in the middle of their range. In this paper (Heijink, 1999), the authors show us that this also applies when playing guitar. So, we have two strands that provide various levels of complexity. The first one in relation to the position of the hand along the guitar neck and the second one in relation to the distance between the fingers. The complexity is lower when the posture is in the middle of the guitar neck and increases the more we move to the left and right neck positions. In relation to finger span, a smaller finger span results in a lower complexity. This is because the position of the hand on a guitar neck is related to the shoulder and elbow joint angles and the finger span is related to the finger joint angles. The joint angles assumed in the postures might have determined the guitarists' complexity ratings.

There are two types of postural transitions that are relevant to mention. The first consists in the necessary transition of the left hand along the neck guitar, the second consists in the permanence of the hand in more or less the same place but which requires a change of the fingers. As we mentioned before, since music is time constrained, and since hand replacement movements are more complex than finger movements, and both have to be done in the same amount of time, we have that the first type is the most complex of the two mentioned above. The authors in this paper (Rosenbaum, 1991) show us that the arm is more suitable for movements of large amplitude and low-frequency while the fingers are more suitable for movements of small amplitude and high-frequency.

In the paper (Heijink & Meulenbroek, 2002), the authors experiment with six professional guitarists, all active performers and teachers graduated from the Brabants Conservatorium in The Netherlands. Each guitarist played a set of 144 sequences. These sequences consist of musical scales of single notes extended upward so that the note sequences became 11 notes in length. In these sequences the postural transitions that were mentioned above were examined. There are four blocks that correspond to small span without hand replacement, small span with hand replacement, large span without hand replacement and large span with hand replacement. Each of these four blocks was played in three different parts that correspond to the low, middle and high hand position on the guitar neck and each guitarist repeats each sequence twelve times. Given that the experiment is performed with professional guitarists and that the sequence of notes consists of playing scales, the cognitive and musical terms are very simple.

In this experiment, each guitarist was asked to provide the easiest fingering for a short note sequence. The large finger span and replacement of the hand along the guitar neck was avoided, as expected. Above we had mentioned that the middle hand position on the guitar neck would be the one that would facilitate a lower complexity. However, it was the low hand position the most chosen guitar neck position. This is possibly due to the familiarity of playing in that zone of the guitar, where it is common to play the base chords on the guitar. Another reason is due to the proximity of the loose notes. A large number of songs are written in E because this scale takes advantage of these same loose notes.

In conclusion the authors understood that the biomechanical bases are important in the choice of positions since they have a direct effect on the complexity of a piece. However, cog-

nitive and musical factors also play an indispensable role in the decision, even if it results in an increase in the complexity of the performance. From a cognitive point of view, a melody that is repeated throughout the music in different pitch heights can be, from a biomechanical perspective, easier to play using different fingerings, each optimizing the instant of the melody in question. However, from the cognitive point of view in these set of melodies may be easier to use a less optimized fingering that fits all the repetitions of the melody. This is because muscular memory has a great weight on the performance of any guitarist. From the musical perspective, a long string generates more harmonics than a shorter one of the same diameter, therefore, if a bright, clear sound is preferred, the string length should be maximized. If, on the other hand, a warmer sound is preferred, then the vibrating string length should be limited.

3.3 Automatic Music Composition

3.3.1 Machine learning to compose tablatures

In this paper (Chen, 2020) is built a model for composing fingerstyle guitar tabs with Transformer-XL (Dai et al., 2019), which is a neural sequence model architecture. This architecture, created as a language modeling architecture, allows learning dependency beyond a fixed length without disrupting temporal coherence, which is quite important in musical events. Given a set of tokens $X = (x_1, ..., x_T)$, the probability of $P(x) = \Pi_t P(x_t|x_{<t})$ is estimated. To address the limitations of a fixed-length context, the authors introduced a recurrence mechanism. So, during training, the hidden state sequence computed for the previous segment is fixed and cached to be reused as an extend context when the model processes the next new segment. With this recurrence mechanism applied to every two consecutive segments of a corpus, it creates a segment-level recurrence in the hidden states.

The authors adapt this architecture to learn how to compose guitar tablatures, where each note is represented as a set of properties such as the duration of the note, the string and the fret, which help model the problem. They want to know if this kind of architecture is able to learn and produce tablatures at the level of those produced by humans. For this, three research questions are proposed: First, will the model be able to learn to generate not only the sequences of notes, but also which fingers to use on each note? Second, Will the model be able to generate compositions with a coherent groove? And third, will the model be able to generate compositions as efficient as those created by humans?

To evaluate the system, they use a dataset of 333 "fingerstyle guitar" tablatures, where thirty tablatures are reserved for validation and only the first 16 bars of each tablature are validated. To answer the first question, the model generates a large number of random event sequences and checks how often plausible strings are generated after the note-on event, which reached a high accuracy. To answer the second question, generated tablatures with and without grooving are compared through a user study, in which the user is shown generated tablatures with grooving and without grooving. Results shows that grooving improves the composition of the tablatures. To answer the third question the generated tablatures are compared with real tablatures through another user study, which shows that the generated tabaltures with grooving are on par with real tablatures.

(Mistler, 2017) thesis on generating guitar tabs with neural networks, approaches the choice of the positions of the notes on the guitar in two different ways. In this work, the problem of

choosing the fingers that press the strings is not addressed.

The first approach is to predict the frets directly. For this, a typical LSTM (Long Short-Term Memory) is used, to model the sequential behavior and predict new elements in the time series, since one can consider the tablature as a time series of guitar frettings used to realize a sequence of chords and notes. The notion of intention into LSTMs is added. In case the pitch sequence is monophonic, i.e., a single note to be played at a given instant, the pitch of time step t is added to the time step $t-1$ so that the fretting output on t depends on the previous frettings and the intended pitch. For polyphonic music, the chord can consist of a maximum of six simultaneous pitches on a 6-string guitar. To model the chords several alternatives have been proposed, so that in the time step preceding the one under evaluation, the information about the chord is available. For the validation of the predicted frettings, there is no guarantee that the predicted ones match the intended chord, so to correct this, that chord is compared with all the possible frettings associated with the chord and the fretting closest to the predicted one is chosen.

The second approach is to predict the frets by cost estimation. The cost function is learned rather than explicitly designed. The cost function is based on the conditional fretting probability, given the context and pitch intention as shown in the equation:

$$P(f_t|f_{t-T},...,f_{t-2},f_{t-1},p_t) = \frac{count(f_{t-T},...,f_{t-2},f_{t-1},p_t,f_t)}{count(f_{t-T},...,f_{t-2},f_{t-1},p_t)} \tag{3.1}$$

And the cost is given by negative log probabilities:

$$c(f) = -\log P_T(f) \tag{3.2}$$

Where $P_T(f)$ corresponds to the equation 3.1.

The cost of the fretting sequences is given by the sum of the individual frettings:

$$c(s) = \sum_{f \epsilon s} c(f) \tag{3.3}$$

Due to the complexity of the fretting space, the calculated probabilities and costs only cover a small fraction of the possible frettings and fretting sequences. To generalize to unseen data, they train a Feed-forward Neural Network to predict the cost of a fretting. The demand for the optimized tab can be seen as the minimum cost of the fretting sequence given by the equation:

$$\min_{s \epsilon S} c(s) \tag{3.4}$$

Where S is the set of all possible fretting sequences.

The dataset used to train and evaluate the system consists of the 100 top-rated Guitar Pro tablatures on Ultimate Guitar[1].

To evaluate the two Machine Learning approaches in the context of the guitar fretting problem, the accuracy measure, and a number of model variations are defined.

[1]https://www.ultimate-guitar.com/top/tabs

The calculation of the accuracy is given by:

$$accuracy(\hat{s}, s) = \frac{1}{t} \sum_{t} \sqcap(\hat{s}_t - s_t) \tag{3.5}$$

where s is a published fretting sequence and $\hat{s}$ is the predicted fretting sequence.

The best baseline for direct prediction used was the sequential model. This heuristic follows the sequential interpretation of chords, which consists in considering the chords as ascending arpeggio, in which each chord consists of a sequence of individual notes. It simply measures the distance between individual note frettings. The generated tablatures from this approach agree with published tablature by 72.9%.

3.3.2 Compose Tablatures by a Set of Rules

In the (Masanobu Miura, 2004) paper, the authors build a system that determines the positions of the notes on the guitar and which fingers to use for each note for melody only. In other words, neither the accompaniment nor the bass is addressed, which entails that only one note is evaluated at any given time. As discussed earlier in section 3.2, there are three factors that determine how preferable a certain note position is on the guitar. In this paper only the biomechanical aspect is addressed. This paper focuses on producing tablatures that decrease the difficulty of the left hand when playing a melody. This work is not only useful for beginners, because when choosing notes on the guitar, even the most experienced guitar players take the biomechanical constant very much into account. As we saw earlier in section 3.2, the movement of the left hand along the guitar neck is more complex than the movement of the fingers keeping the position of the hand in the same place on the guitar neck. Given this, for the complexity calculation, this paper tries to minimize the movement of the left hand along the neck by implying a larger movement of the fingers.

The S2T, "Score-to-Tablature" system they created, divides the melody into phrases, which consist of a set of sounds close together in time. The user chooses the position of the first note and then searches for a combination that does not require movement of the left hand among the possible combinations of finger positions for each sound in a phrase. In specific terms, the difference between the highest fret and the lowest fret used in a phrase is defined as "the required finger position span." A fingering system which minimizes the required finger position span is created. When there are several sequences that minimize the required finger position span, the position for the left hand which includes the starting finger position input by the user for the first phrase is used. When there are multiple candidates for the first phrase, the one with the smallest amount of overall movement for the left hand in the piece overall is used. After the note positions are chosen, if the required finger position span in a phrase is less than four frets, the determination of the fingers is done using a one-to-one correspondence between the frets and fingers. If five or six frets are needed, the two frets at the end are assigned with the index finger or little finger. However, playing two successive notes with these fingers is difficult, so the middle finger or ring finger is used.

In order to evaluate the system, two points of view were considered: the amount of movement of the left wrist between phrases and the required finger span in a phrase. For the first point of view, the output results were analyzed for the melody part (vocal part) of the J-POP8 scores. The authors conclude that the movement of the left wrist between phrases is kept to a

necessary minimum. Then they evaluate the system by comparing it to two other commercial systems for the melody described above, which shows that this system can produce tablatures that are played easily, which is appropriate for guitar novices.

4 Method

4.1 Introduction

Here we explain our system, GuiTab created to produce a tablature receiving as input a single guitar MIDI file, where the system finds the most optimal way to play the desired notes while also assign a specific finger to press the fret to produce the desired note. We can say that the system we propose is inspired in the system created by - (Masanobu Miura, 2004). However, unlike their system, we do not focus only on the melody, but on three parts of the music: the melody, the accompaniment and the bass. That is, we do not consider only monophonic music, but polyphonic music. We also use the concept of phrases explained above, however, with a different division. The finger association at its base is the same, but the complexity is much greater since we have to associate fingers with chords and barre chords.

Our motivation to develop a system based on heuristics and rules that restrict and validate the hypotheses of a set of notes played in certain positions with certain fingers is due to three factors: the first factor comes from the idea that when a human reads a score and passes it to the guitar, he relies on a set of rules (which he may not even be explicitly aware) to decide how to play the score. For a large set of notes, there are millions of different possibilities to play those notes, and yet, a professional and experienced guitar player can in a short period of time define the best or one of the best ways to play them. Given this, there are certainly sets of rules, some basic, some not so basic, by which a guitarist is based.

The second reason is due to the advantage of using this technique when evaluating unknown set of notes as opposed to machine learning systems. Since there are few tablatures available to train the machine learning systems, it is recurrent the existence of sets of notes that are unknown to these systems.

Lastly, this technique does not require a dataset of tablatures, to be able to create a system capable of composing tablatures in opposition to using machine learning where a large set of tablatures is needed. Since for a particular music there may not be a best tablature but rather a set of several possible tablatures, the dataset would have to be even larger to create a robust system. Moreover, the system requires tablatures that with errors so that the system can be more aware of what can and cannot be done in the association of positions.

4.2 Requirements and Features

The machine we use to run our system had an Intel Xeon CPU at 2.40GHz, 48GB of RAM and was running CentOS Linux 8. The run time required to create the output of a song with an average of 1000 notes was 3 days. However, in addition to the number of notes in each music, it is necessary to consider dependencies between notes. When there are dependencies

between notes, there are more restrictions in the construction of the hypotheses. Songs with fewer dependencies between notes generate many more hypotheses and the run time is much longer. Given the memory capacity of our machine, it is not possible to keep all paths during compilation. During our tests, there were songs that reached 70 million different paths. Given the number of possible paths, we had to perform cuts on the least promising paths. The process of path exclusion is explained in the section 4.6.

4.3 Architecture

In this section we present the architecture of our system *GuiTab* and, in a not so detailed way, explain how the system works. The architecture of the system is shown in figure 4.1. In MIDI processing there is MIDI extraction and a set of processing phase in order to prepare the set of notes extracted from MIDI for our system. In Hypotheses Generation the hypotheses are generated, they are nothing more than a set of notes with defined positions. In other words, each hypothesis is one of the possible ways to play a set of notes. Here it is also checked whether or not it is possible to play the hypothesis. In Hypothesis Selection, hypotheses which are difficult to be executed are excluded, and a set of heuristics is used to rate the paths of the various hypotheses and the best path is chosen.

In fingers optimization the choice of fingers associated with each note of the best rated path are made. Finally, in Tablature Creation the tablature of the best path is created.

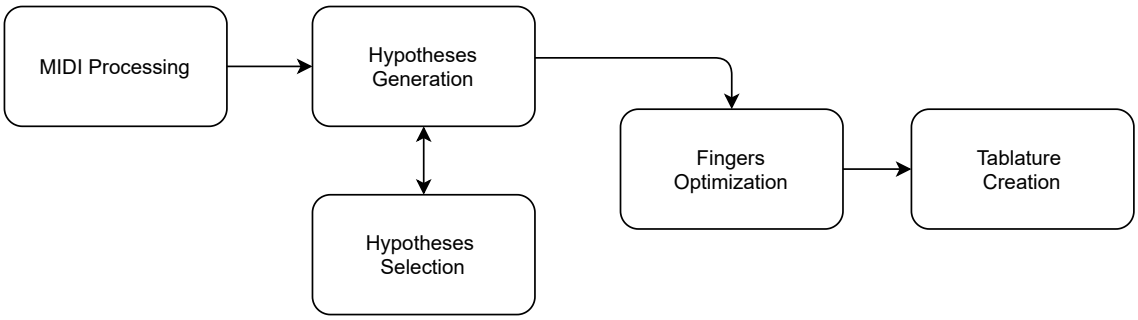

Figure 4.1: Architecture

4.4 MIDI Processing

This section explains how we extract and process MIDI files, our system's input, and how it is passed to the base for our system.

4.4.1 MIDI Extraction

For extracting the MIDI file we use the python library Music21[1]. When we extract the MIDI file, a score is created which consists of a set of three channels. Each channel has information about the instrument used, the time signature and has a set of notes that correspond to each part of

[1]http://web.mit.edu/music21/

the music. The first channel contains the bass notes, the second channel the accompaniment notes and the third channel the melody notes. Given that, we have to sort the notes in relation to their appearance in the piece, each note resulting from MIDI extraction consists of an object of the Note class present in the Music21 library with a set of attributes such as the offset of the note, the onset of the note, the duration of the note, the pitch of the note and to which octave it belongs. We then sort the notes by the onset associated with each note. If several notes have the same onset, the notes are sorted from lowest to highest pitch. If we sort the notes present in figure 4.2, we have the following set: [E2, B4, B3, G3, B4, B3, G3, B4, B3, G3].

Finally, since the MIDI file is divided into three channels corresponding to bass, accompaniment and melody respectively, we need to eliminate repeated notes. Let us observe figure 4.2 where the initial bar of the song Spanish Romance (Spanish Romance, 2017) is present. If we look at the first high note, we see that it is in theory part of both the melody and the accompaniment, because when we look at the note in question, we see that it has both a dash upwards, which shows that it belongs to the melody, and a dash downwards, which shows that it belongs to the accompaniment. So, when the MIDI is extracted, this note appears both in the channel that contains the melody notes and in the channel that contains the accompaniment notes. As in practice there is only one note, we do not want to consider it twice. One of the notes is then eliminated, the note which has less duration.

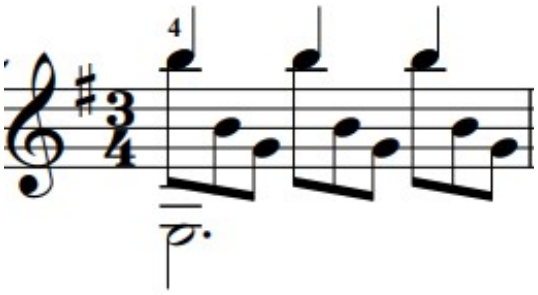

Figure 4.2: Spanish Romance

4.4.2 Notes Division

After we have the notes sorted as explained above, we divide them into subsets, or phrases, using the same nomenclature as used by (Masanobu Miura, 2004). However, the division we propose is different and more appropriate for our purposes, since our system evaluate polyphonic music, and therefore there are dependencies between notes. We divide the notes taking into account the dependencies between them. We consider two types of dependencies: direct and indirect dependency. When two notes are directly dependent it means that they are both heard in one or more instants of the music and two notes are indirectly dependent when they are not directly dependent, i.e., they do not coexist in one or more instants of the music but have at least a third note in common which is directly dependent on these two. Looking at the picture 4.3 two directly dependent notes are for example the first two notes in green, in fact, they have the same onset. Two indirectly dependent notes are for example the two green notes at the top, since they are independent from each other but dependent on the third green note.

Therefore, a new subset is created when a new note is independent of the notes belonging

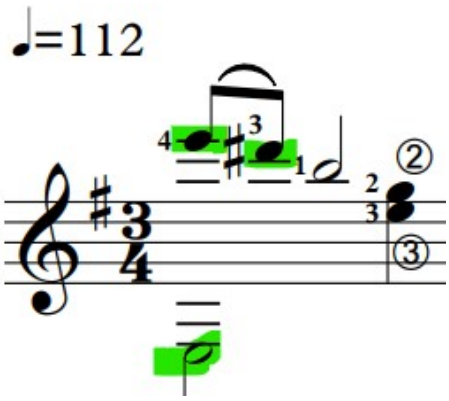

Figure 4.3: Excerpt of Adelita.

to the previous subset. If we think about the physical aspect, i.e., the physical limits of the left hand when we have to press a set of frets to play a set of notes, when these are dependent, it means that several fingers have to press a number of notes at the same time instants. If we consider two dependent notes where one is played at fret one and the other at fret twelve. It is impossible to perform this set of notes in these two positions, since the left hand cannot press a string at fret one with one finger and another string at fret twelve with another finger. So when we divide the sets in this way, we are on the one hand constraining the distance between notes within the set and, on the other hand, releasing that constraint between the sets.

Then we require the creation of the concept of moment. The moment is the instant of time when one or more notes appear in the music. Looking at figure 4.2, we have nine different moments, because we have ten different notes where two of them have the same onset. The first two notes start at moment 1, and when one or more notes with a different onset appears in the music the moment is increased by one. This concept is then added to each note, where for each note we add the number of moments where this note is heard in the music, i.e., the note is associated with all numbers corresponding to the moments when that note is sounding. In composition from figure 4.2, the first lowest note lasts the whole set and therefore its set of moments is from moment 1, when the note starts to be heard in the music, to moment 9. This concept makes it easier to check for direct dependencies between notes. If two notes have at least one moment in common they are directly dependent on each other, because at that moment, both notes are heard in the music. Then, the subset is further divided into subsets taking into account the moments at which each note starts in the song. This way notes which start at the same instant of time belongs to the same subset. So, the notes are divided into subsets taking into account their dependencies, which in turn is divided taking into account their musical onset. We explain later in section 4.5 the reason of this second division.

We also need to deal with trills. A trill is a musical ornament that consists of the rapid alternation of, usually, two adjacent notes, semitone or tone apart. A trill has no tempo. As an example in figure 4.4, we have a bar from the piece *Adelita* by *Francisco Tárrega*. This bar is a 3 by 4 bar, i.e. it consists of three notes of quality 4 (crotchet) or their corresponding notes. If we look at the top notes, the ones that correspond to the melody, ignoring the first two, we have 6 quavers that correspond to 3 crotchets, and we have a note present at the bottom, corresponding to a minim with a dot, which also makes 3 crotchets. This means that the first two notes correspond to a trill and do not enter into the counts, in terms of time, of the bar.

They are just a musical ornament, when we extract notes corresponding to this ornament, their duration is 0, because, in fact, in the bar context their duration does not exist.

However, in order to deal with the dependencies between notes and their duration throughout the music, we cannot have notes with duration 0. These first two notes not only have duration 0, but their onset corresponds to the same onset as the next two notes. Notes that start at the same onset are directly dependent. If we look at the gold standard of this piece, we have that all three melody notes are played on the same string, and we know that directly dependent notes cannot be played on the same string. This means that even though they start on the same time instant, they are not dependent on each other. To solve this, for each set of dependent notes we check if there are notes of this type, the duration of each note with duration 0 is changed to 0.0001 so that they have some duration but not enough to change the bar structure, and the onset is changed too. In this example, for the second note of the trill the onset decreased by 0.0001 and for the first note of the trill the onset is decreased by 0.0003, so that these three notes start at different times, but do not change the bar structure.

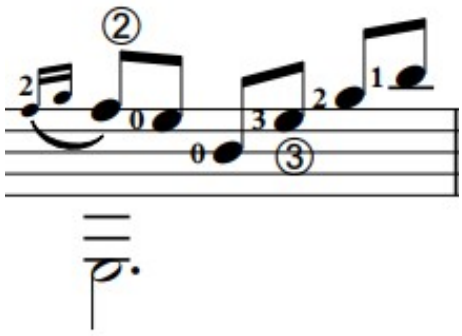

Figure 4.4: Adelita

4.4.3 Tune Verification

Finally, we check which tuning the music is in. Our system only considers two different tunings. The standard tuning (E-A-D-G-B-E) and the drop D tuning in which we lower the sixth string by one tone from the standard tuning (D-A-D-G-B-E). We also chose to include this second tuning because it is quite common in classical music pieces. So, we have two abstractions of the guitar neck, one for each tuning. In fact, only the positions of the notes of the sixth string change depending on the tuning. To check if the tuning is in Drop D, we just check for the presence of *D2* or *D2#* notes, which are the only notes that do not exist in E tuning and become present in this one.

4.5 Hypotheses Generation

At this point our system generates possible hypotheses and checks if they are valid, i.e., if they correspond to possible playing notes. As mentioned above, a hypothesis is a set of notes with certain positions on the guitar. A path is a set of hypotheses that together make up the music being evaluated. In figure 4.5, it is possible to see the Hypotheses Generation architecture.

Earlier we talked about how the splitting of notes is performed. We divided them into sets based on their dependencies and into subsets based on their onset in the music. Looking at figure 4.2, as an example, all the notes in this bar belong to the same set and are then divided into nine subsets.

The way we generate the hypotheses is incremental, which means that we take the first subset and generate the possible hypotheses for the notes in that subset, then we move to the next subset, we generate the hypotheses considering the notes in the previous subset and the present subset, and so on. This means that the notes are evaluated repeatedly. However, we realized that doing it incrementally is much more efficient. If, when we take the first subset and generate the possible hypotheses, some hypotheses are invalid, i.e., there is no possible way to play that set of notes in the given positions, and they are excluded. By invalidating hypotheses, we are cutting off paths which should not be explored further. Suppose that in the first subset we generate 10 different hypotheses and in the next subset we generate 6 different hypotheses. The set of possible paths, given these two subsets, is the Cartesian product between the hypotheses of the first subset and the hypotheses of the second subset which are 60 possible paths. Doing the process incrementally we take the first subset and check which hypotheses are invalid. If four of the ten hypotheses are invalid, the number of possible paths given the two subsets is the Cartesian product between the six valid hypotheses of the first subset and the six hypotheses of the second subset, which are 36. The incremental process, therefore, allows us to cut hypotheses earlier and makes the exploration of possible paths more efficient. If, instead of the incremental process, we took all the notes in the set and tried to generate the possible hypotheses with all the notes, the number of paths to explore would be exponentially large.

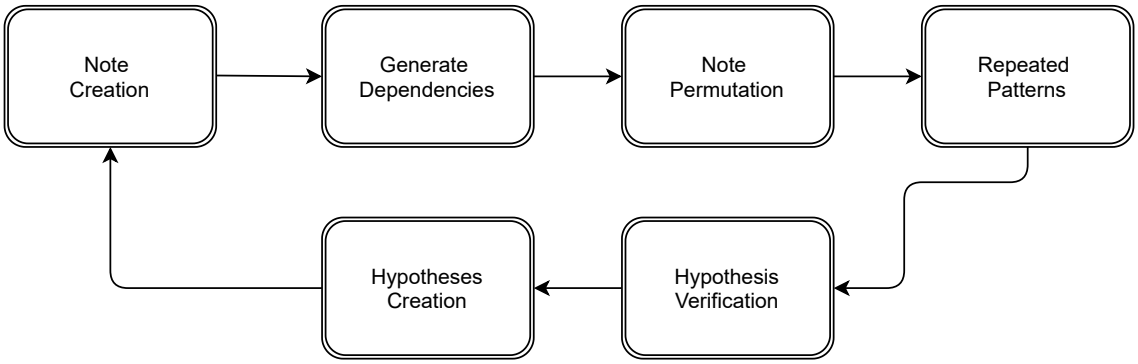

Figure 4.5: Hypotheses Generation Architecture

4.5.1 Note Creation

After creating the sets as explained above, we move on to creating the notes for our system. For now, the notes belong to the Note class of the Music21 library. However, because we need to add extra attributes to the notes, we created ourselves the super class Note which has as sub classes all the notes we consider. That is, there are 46 sub classes, from the lowest note, D2, to the highest note, B5. The super class has a set of attributes common to all notes while the sub classes have specific attributes to each note.

Suppose we want to create the notes in red present in figure 4.6. To create the notes E2 and B3, we have to figure out in which positions both notes exist through our abstraction of

E4	F4	F#4	G4	G#4	A4	A#4	B4	C5	C#5	D5	D#5	E5	F5	F#5	G5	G#5	A5	A#5	B5
B3	C4	C#4	D4	D#4	E4	F4	F#4	G4	G#4	A4	A#4	B4	C5	C#5	D5	D#5	E5	F5	F#5
G3	G#3	A3	A#3	B3	C4	C#4	D4	D#4	E4	F4	F#4	G4	G#4	A4	A#4	B4	C5	C#5	D5
D3	D#3	E3	F3	F#3	G3	G#3	A3	A#3	B3	C4	C#4	D4	D#4	E4	F4	F#4	G4	G#4	A4
A2	A#2	B2	C3	C#3	D3	D#3	E3	F3	F#3	G3	G#3	A3	A#3	B3	C4	C#4	D4	D#4	E4
E2	F2	F#2	G2	G#2	A2	A#2	B2	C3	C#3	D3	D#3	E3	F3	F#3	G3	G#3	A3	A#3	B3

Figure 4.6: Notes Creation Example.

the guitar neck. The note E2 only exist in one position: (6,0), while the note B3 exists in five positions: (2,0), (3,4), (4,9), (5,14) and (6,19). So, when we create notes, we create n different objects of the same note, only varying the position of each object. So, for note E2 only one object is created, since this note is only in one position, and for note B3, five objects are created. Each note object is also associated with a shared id. The id is shared between the objects of the same note. So, the object in note E2 has the id number 1, while the five objects in note B3 have id number 2. Because objects of the same note are parallel objects, they are never be tested together since they belong to the same note.

Before creating the notes, we changed flat notes to the corresponding sharp or natural notes. In theory a B2 flat is different from an A2 sharp. However, in practice they are the same note with the same frequency. In our system we do not recognize flat notes, so we pass them to the corresponding notes, either sharp or natural.

4.5.2 Generate Dependencies and Cartesian product

We need to generate dependencies between notes. Each object note has a list of the notes it is directly dependent on, that is initialized empty. When generating dependencies, a single object from each note is used to calculate the dependencies. To calculate the dependencies, we just have to check the duration of each note and see if they both have at least one moment in common, if they do, it means that at least in that moment they are heard momentarily and therefore are directly dependent. Each list in this object is filled with the dependent note ids, which is then propagated to the other objects of the respective note.

After generating dependencies between the notes, the Cartesian product is made between the notes of the subset under evaluation. Consider that in the first subset we have notes E2 and E5 which after created are presented as two lists, where in the list for note E2 there is only one object which corresponds to position $(6, 0)$ while for note E5 there are two objects, which correspond to positions $(1, 12)$ and $(2, 17)$. Therefore, the Cartesian product is as follows: $[(E2, E5), (E2, E5)] = [((6, 0), (1, 12)), ((6, 0), (2, 17))]$. We can observe that the Cartesian product corresponds exactly to the different ways of playing a set of notes, which we call hypotheses.

Suppose that both hypotheses are valid and we move on to the next subset which corresponds only to note B3. Therefore, we create the note B3 which results in 5 objects corresponding to the positions $(2, 0)$, $(3, 4)$, $(4, 9)$, $(5, 14)$ and $(6, 19)$. After creating the notes and generating the dependencies between this note and the two previous notes we generate the new possible hypotheses. This also results in a Cartesian product between the B3 note positions and the previous two generated hypotheses, where the result is as follows:

[((6,0),(1,12),(2,0)), ((6,0),(1,12),(3,4)), ((6,0),(1,12),(4,9)), ((6,0),(1,12),(5,14)), ((6,0),(1,12),(6,19)), ((6,0),(2,17),(2,0)), ((6,0),(2,17),(3,4)), ((6,0),(2,17),(4,9)), ((6,0),(2,17),(5,14)), ((6,0),(2,17),(6,19))]. With only three notes, where one has one position, one has two different positions and one has five different positions, the number of possible hypotheses is 10. The growth of possible hypotheses is exponential, and this underlines our idea of creating the hypotheses incrementally. Some of these hypotheses are not valid and if they are not valid, their paths should not propagate, which results in a decrease in the growth of possible paths.

4.5.3 Different Cases in creating Hypotheses

As mentioned above, hypotheses are the possible ways to play the set of notes being evaluated. There are four different cases in generating hypotheses. In figure 4.7 we can see the process of the first and second case. The first case is when we are evaluating the first subset and that subset contains only one note. In this case the hypotheses created are only the various positions of this note. If we only have to play one note, there are no restrictions on playing any of its positions. So, if we are in the first case, and the note has 3 different positions, three hypotheses are generated each containing one of the three note objects. The second case is when we are evaluating the first subset and that subset contains more than one note. Since there are more than one note, it is necessary to check the dependencies between notes. The generated hypotheses are the result of the Cartesian product between notes. If we have two notes: note E2 with one position (6,0) and note E5 with two different positions (1,12) and (2,17), the generated hypotheses are 2. The first hypothesis is: h1:[E2: (6,0), E5: (1,12)] and the second hypothesis is: h2:[E2: (6,0), E5: (2,17)].

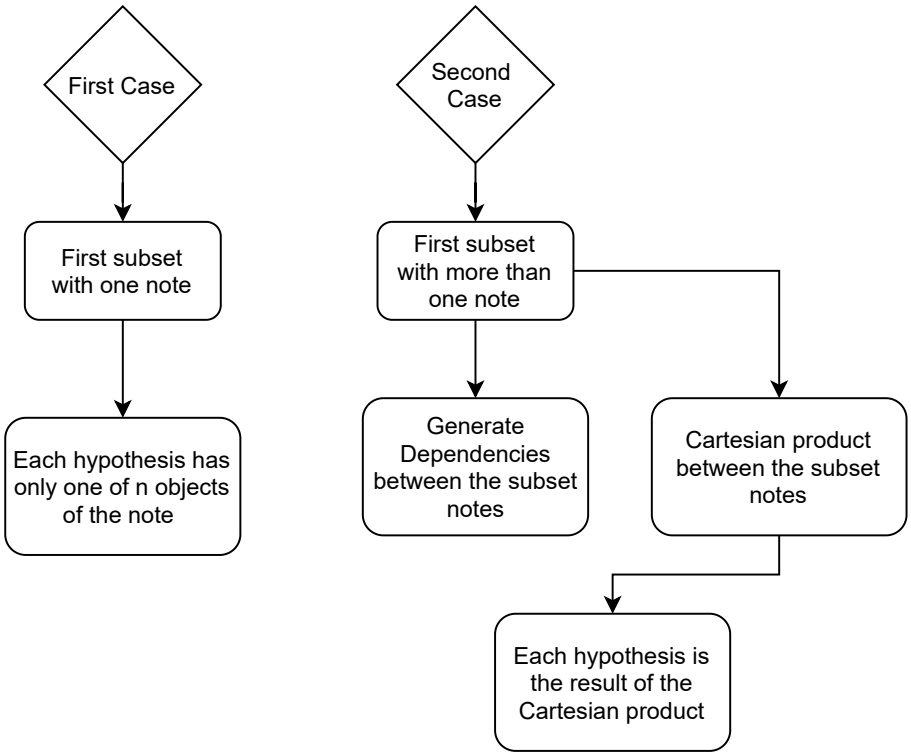

Figure 4.7: First and second cases.

In figure 4.8 we can see the process of the third case. The third case consists of the case where we are no longer in the first subset and the present subset contains only one note. We mentioned earlier that the way we generate the hypotheses is incremental, which means that there are already hypotheses generated from the previous subsets. Suppose that there are 3 hypotheses generated from the previous subsets, and that the note from this subset have two different positions. The new generated hypotheses consist of the Cartesian product between the hypotheses already generated and the new note positions as seen in section 4.5.2.

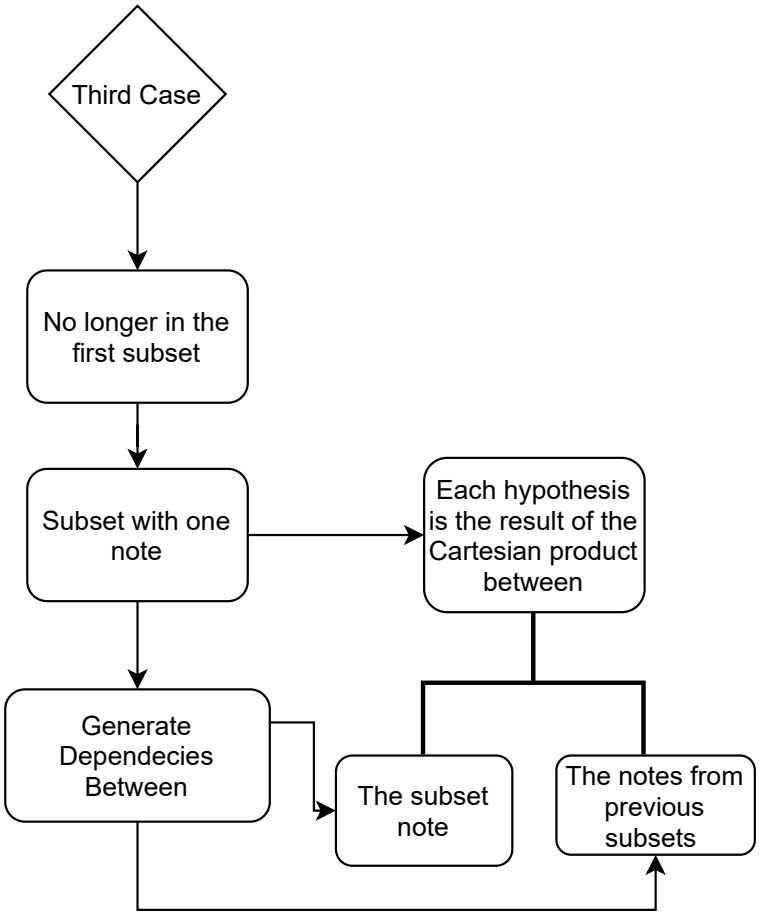

Figure 4.8: Third case.

Finally, in figure 4.9 we can see the process of the fourth case. The fourth differs from the third case in that the new subset contains more than one note. We have to generate the dependencies between the subset notes and between the subset notes and the notes from the previous subsets. Since the subset contains more than one note, it is necessary to make the Cartesian product between the objects of the notes from the subset and then make the Cartesian product with the hypotheses already generated. Suppose that in the present subset we have two notes, both with two different positions: first note: $[position_{11}, position_{12}]$, second note: $[position_{21}, position_{22}]$. The Cartesian product between these notes are the following: $[position_{11}, position_{21}]$, $[position_{11}, position_{22}]$, $[position_{12}, position_{21}]$ and $[position_{12}, position_{22}]$. Suppose that there is 1 hypothesis generated from the previous subsets. The new generated hypothesis is 4, resulting in the addition of the result of the previous Cartesian product to the previously generated hypothesis.

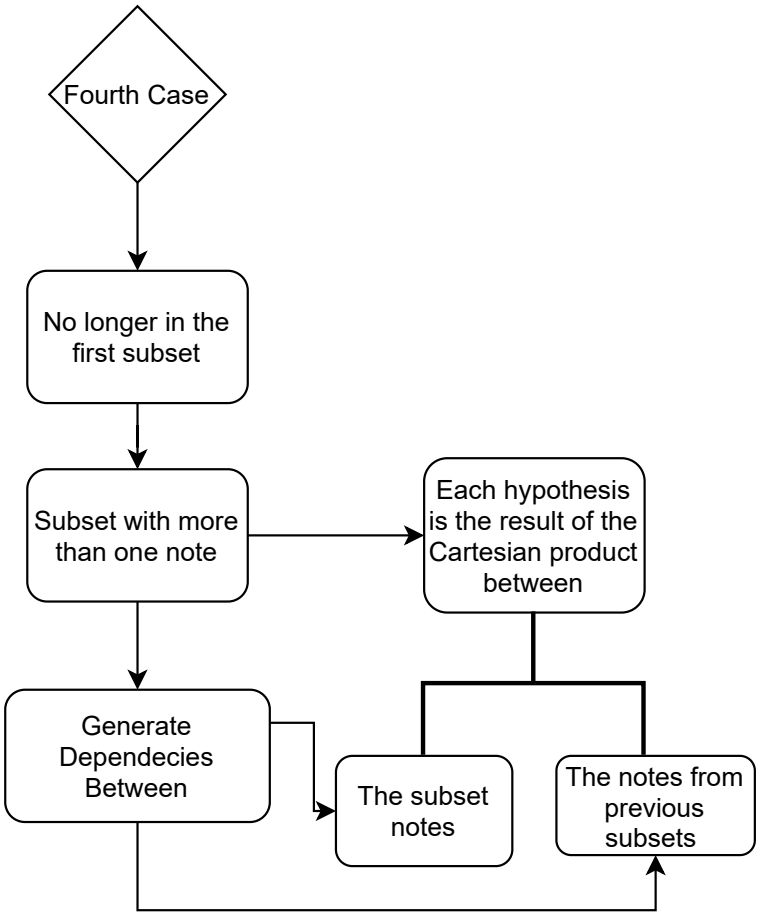

Figure 4.9: Fourth case.

4.5.4 Repeated Notes

In the 2.1.1.3 section we mentioned how to check for repeating notes, that allows us to exclude some hypotheses before we even evaluate whether they are valid or not. In figure 4.10 there are 3 notes that repeat throughout the set. The note E5, the note B3 and the note G3. It is in the repetition of notes within a set that there is this verification. Pretend that the first note of each of them has already been evaluated and the hypothesis in question is as follows: [E2:(6,0), E5:(1,12), B3:(3,4), G3:(3,0)]. The next note to be evaluated is note E5 for the second time. Are we interested in evaluating this hypothesis [E2:(6,0), E5:(1,12), B3:(3,4), G3:(3,0), E5:(2,17)] or only these two: [E2:(6,0), E5:(2,17), B3:(3,4), G3:(3,0), E5:(2,17)], [E2:(6,0), E5:(1,12), B3:(3,4), G3:(3,0), E5:(1,12)]?

By doing this we are stabilizing the left hand which only has to repeat the previously played position, thus optimizing the choice of positions. First, we have to check if the same position can be used. This may not be possible when there is another note which is directly dependent on the new note and is on the same string, or when the same note played earlier in the set is dependent on this new note. When this happens, we know that the note cannot have the same position. If it is possible to use the same position we see if we should exclude positions that are different from the already used one. If the first note is a loose note and the

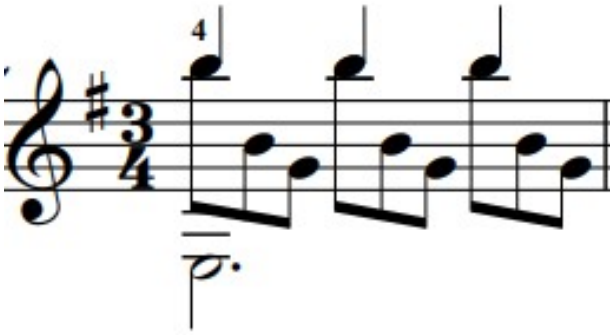

Figure 4.10: Excerpt from Spanish Romance.

same position can be repeated, we exclude all the different positions. For example, in the case of the note G3 in our example that has position (3,0), new G3s notes that are not in position (3.0), are excluded in the case the first G3 is in position (3.0).

Here we do not completely exclude all other positions, we just keep it consistent when possible. There may be hypotheses where the position of G3 is (4,5), (5,10) and (6,15). We are only trying to keep the same position when the note appears more than once in the set. If the first note is not a loose note and the new note we want to add is, we use the heuristic given by the following equation:

$$heuristic = \sqrt{(fret_1 - fret_2)^2 + (string_1 - string_2)^2} \tag{4.1}$$

This heuristic calculates the difference between the frets and the strings of the two notes. If the result of the heuristic is less than a threshold the new position is excluded, otherwise it is not excluded.

If none of the notes are loose notes, as in the case of the repeating note E5 in our example, we calculate the heuristic shown in equation 4.1 between the position of the first note and the current location on the guitar neck and the distance between the new position and the current location on the guitar neck. If the first result is less than or equal to the second result, we exclude the new position. To conclude, we are checking whether we have moved far from the position where the first repeated note was played to see if it is worthwhile to take hypothesis where the position changes or not. This check is only done in cases 3 and 4, because these are the cases where there are already evaluated notes.

4.5.5 Hypotheses Verification

At this step, the hypotheses are verified, that is, of all the possible hypotheses we check which are valid and which are not. There are two types of factors that can and cannot validate a hypothesis. Some make a hypothesis impossible, such as two dependent notes being at a great distance from each other, or dependent notes being played on the same string. Other factors as we have seen in section 4.5.4 help us understand whether or not it is worth pursuing the respective hypothesis.

Now we evaluate the hypotheses that have not been excluded. We have two factors to work by when evaluating the hypotheses. The first factor is the distance between notes and the second factor is whether all notes can have associated fingers. In fact, the distance between notes has a lot to do with whether we can associate fingers with notes, because notes at a great distance create a physical impossibility of being played. However, there are other reasons why it may be impossible to associate fingers with all the notes as we show below.

4.5.5.1 First Finger Association

This first association of fingers to each note is not the final association one. This is because in this first phase of validating or not validating hypotheses, each group of hypotheses does not take into account its context, i.e., the other groups of hypotheses. One can imagine that each group of hypotheses constitutes all we know about this piece of music, and therefore the choice of fingers only takes into account the notes in that group. To better understand the difference, we look at the following example: in the figure 4.11 there are three sets of a single note, since each note is independent from the next. As we described above, when we associate the fingers to the notes of each set, we do not take into account either the next or the previous sets. So, which finger would be associated in the validation of each hypothesis? And which finger would be associated to each note already taking into account its context?

Finger choices when playing a set of notes are subjective, and guitarists have different tastes and preferences. The different physical characteristics of each guitarist mean that what may be easy to perform for one guitarist may be extremely difficult for another. For example, a guitar player with long fingers finds it easier to travel longer distances between notes. However, there are certain rules in finger picking that are general to all guitar players. Looking at figure 4.11, when we only have one note to associate, we always associate it with finger 1, the index finger. Since we know nothing of the previous and subsequent notes and as we talked about earlier, each set at this stage is all we know of this piece of music, if a set has only one note, this note is associated with the most common finger, which is usually the strongest. So, as an answer to the first question, all three notes would be associated with finger 1. When the context is analyzed, does it make sense to play all notes with the 1st finger? No, having four possible fingers to use and a very restricted time to play each note, it does not make sense to always use the same finger on notes with different positions. So, as an answer to the second question: no, the fingers that would be associated at the end would not always be the 1st finger. However, without knowing the context it is not possible to know which finger we should choose.

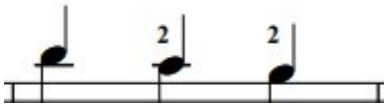

Figure 4.11: Spanish Romance

4.5.5.2 Hypotheses Division

Sometimes we need to divide the hypothesis into small sets. This is needed when the note that connects the whole set is a loose note. We said earlier that one of the reasons ensembles are

defined this way is because notes which are dependent directly or indirectly force less breadth in the use of the guitar neck. However, when the note that connects the set is loose, i.e., no finger has to be used to press the fret of the guitar to hear it, this allows us a greater range within the set itself. In figure 4.10, we see that the note E2:(6,0) links the whole bar, because it is dependent on all the notes in the bar. So, in that case we need to divide the hypothesis into small sets. Suppose that one of the hypotheses that has to be evaluates is the following: [E2:(6,0), E5:(1,12), B3:(4,9), G3:(3,0), E5:(1,12), B3:(4,9), G3:(3,0)]. To do this division, a set is created with the non-loose notes, which in our example would give the following set: [E5:(1,12), B3:(4,9), E5:(1,12), B3:(4,9)]. Then the division of this new set is done in the same way as the division of sets described in section 4.4.2, which in this example the division would therefore be: [E5:(1,12), B3:(4, 9)] and [E5:(1,12), B3:(4, 9)].

4.5.6 Validation Process

In the validation process we try to associate fingers with all the notes of the hypothesis. For a hypothesis to be valid we check if all notes that belong to the hypothesis have fingers associated with them. If they do not, the hypothesis is not valid and is excluded. For case 1, where there are no previously evaluated notes yet and there is only one note to be evaluated in this subset, all hypotheses, which have only one note in different positions, are valid and for positions where the fret is not 0 the finger 1 is associated and 0 for where it is.

For the other three cases the validation is more complicated. To follow the validation process, we take the example present in figure 4.10 and we consider that the new note to be evaluated is the seventh note, which corresponds to a G3. One of the hypotheses that has to be evaluated is the following: [E2:(6,0), E5:(1,12), B3:(4,9), G3:(3,0), E5:(1,12), B3:(4,9), G3:(3,0)]. Each hypothesis has a reference note, or if the hypothesis has been divided into subsets, each subset has a reference note. The reference note is the note with the leftmost fret in each set and if there are notes on the same leftmost fret it is the note on the lowest string. In our example, the hypothesis is divided into two subsets: [E5:(1,12), B3:(4, 9)] and [E5:(1,12), B3:(4, 9)], and for each subset the reference note is the note B3. We start by associating the finger to the loose notes, i.e., these three notes in our example: [E2:(6,0), G3:(3,0), G3:(3,0)]. We start with the note E2 and check if there is any note dependent on it on the same string that has already been evaluated (which at this point was none). If there is not one, we assign a finger ”0” and add it to the notes already evaluated. We do the same for the remaining loose notes. Both G3 notes have the same position, but since they are independent of each other it is possible to play them in the same position. If there is one note dependent on the same string, no finger is associated and the finger attribute stays with the string it was initialized with, ”Undefined”.

Next, we associate the fingers with the reference notes. Each reference note is associated with finger 1. Within each subset it is the leftmost note, we want to use finger 1. For obvious reasons, notes that are further to the left than others have smaller fingers than notes that are further to the right. Looking at these two positions, (4,9) and (1,12), it would not make any sense to associate finger 1 to the second position and finger 2 to the first, nor finger 2 to the second position and finger 3 to the first, and so on. Here too the same check is made as in the case of the loose notes. In our example, so far we have that the notes evaluated are: [E2:(6,0) finger: 0, B3:(4,9) finger: 1, G3:(3,0) finger: 0, B3:(4,9) finger: 1, G3:(3,0) finger: 0].

4.5.6.1 Finger Estimation of the Remaining Notes

Whatever the hypothesis to be evaluated the first notes evaluated are the loose notes and the reference notes. And the process is always the same, the loose notes are associated with finger "0" while the reference notes are associated with finger 1. For the remaining notes the finger associated depends on the distance in terms of frets between the evaluated note and the respective reference note.

Considering the previous example, only these two notes remain to be evaluated: [E5:(1,12), E5:(1,12)]. For each note we need to find its reference note, and from that one we associate a finger to this one. For our example each note E5 is associated with each note B3, given the separation of the above sets. After finding the respective reference note and check that there is no note already evaluated that is on the same string as the one being evaluated the distance between the frets is calculated, which is given by the following equation:

$$PossibleFinger = abs(fret_{ref} - fret_{eval}) + 1 \tag{4.2}$$

This calculation results in the first estimate of the finger we want to associate with the evaluation note. The *PossibleFinger* for the E5 note is: abs(9 - 12) + 1, which is equal to 4, where 9 is the fret of the reference note and 12 is the fret of the evaluation note. To understand why we add another 1 to the difference between the frets, just think of two contiguous notes in relation to the frets where the first note is the reference note. The distance between frets is 1. However, the reference note is already associated with finger 1 and so we want to associate it with the next finger, 2.

After the first estimation of the finger, we check if it is possible to match it. We decided to divide the guitar neck in 4 parts, because the distance between the frets decreases from left to right. This makes the frets that the fingers can travel vary along the arm. The first part covers frets 1 and 2, the second covers frets 3 to 7, the third from 8 to 12, and the last from 13 to 19. To check which part of the guitar neck we are in, simply check the fret of the reference note associated to the note that is being evaluated. In this example we would go to the third part since the fret of the reference note is 9. The difference between the various parts of the guitar neck is the maximum value that the result of the equation 4.2 can have.

The result of this equation can be seen from two different angles, the first angle is which finger we can use for the note we are evaluating, while the second angle is the real calculation of the equation, the distance between the two notes in relation to their frets. If the distance of the frets varies depending on the different sections, the maximum distance that the fingers can travel also varies. The greatest distance of the frets that the hand can travel is possible by using finger 1 and finger 4, as these are the fingers at the ends. It must be taken into account that given the variation in hand and finger size of each guitarist, these thresholds can be small or large for certain guitar players, so we have tried to put a value that corresponds to the distance that a normal sized hand can travel. So, the system allows system users to change these values as they wish. The default values are: for part one the maximum distance between the fret of the evaluation note and the reference note is 5, for part two it is 6, for part three it is 7 and for the last one it is 8. If the maximum distance is exceeded, no finger is associated with the note being evaluated.

If the maximum distance is not exceeded, the distance of the strings between the two notes is checked. Let us look at the following example in the figure 4.12. In this example we consider

the notes colored red. When calculating the distance between F#2 and G4 given by the equation above, the *possibleFinger* is 2. However, given the large distance between the strings, using finger 3 for G4 would be far more efficient and easier to perform. This distance between strings is only checked when the finger movement is downward, i.e., when the string of the reference note is above the string of the note to be evaluated. Therefore, 1 is added to the initial value of the distance between the two notes if the hypothesis in question is not a barre. The *possibleFinger* is increased by 1 if the distance between the strings is at least 3, and for the remaining parts, if it is 4. Suppose that besides notes F#2 and G4, we also had notes C#4, A3 and E3 on the same fret as F#2 present in the hypothesis. All notes of the 3rd fret would be associated with finger 1 forming a barre as explained in section 2.1.4. The difficulty of execution present in the previous hypothesis would disappear, since finger 1 is already close to the position of the G4 note.

In our example from figure 4.10 B3:(4,9) and E5:(1,12), given the distance between the strings, *possibleFinger* is not increased by 1. After we have defined the first estimation of the finger to use, we check if it is possible to associate it, or if another one has to be associated, or if it is possible to associate any finger at all. The result of the equation 4.2 can be greater than 4, so the *possibleFinger* is the minimum between the own value and 4, which is the last finger value.

E4	F4	F#4	G4	G#4	A4	A#4	B4	C5	C#5	D5	D#5	E5	F5	F#5	G5	G#5	A5	A#5	B5
B3	C4	C#4	D4	D#4	E4	F4	F#4	G4	G#4	A4	A#4	B4	C5	C#5	D5	D#5	E5	F5	F#5
G3	G#3	A3	A#3	B3	C4	C#4	D4	D#4	E4	F4	F#4	G4	G#4	A4	A#4	B4	C5	C#5	D5
D3	D#3	E3	F3	F#3	G3	G#3	A3	A#3	B3	C4	C#4	D4	D#4	E4	F4	F#4	G4	G#4	A4
A2	A#2	B2	C3	C#3	D3	D#3	E3	F3	F#3	G3	G#3	A3	A#3	B3	C4	C#4	D4	D#4	E4
E2	F2	F#2	G2	G#2	A2	A#2	B2	C3	C#3	D3	D#3	E3	F3	F#3	G3	G#3	A3	A#3	B3

Figure 4.12: Distance between strings example.

4.5.6.2 Finger Estimation Validation

After obtaining the most suitable finger to associate with the note to be evaluated, we check if it is possible to associate it. In this verification two different paths are followed. The first happens when the note to be evaluated is not in the same fret as the reference note and the second when it is. The idea is to see if of the notes that have already been evaluated, any forbids the use of the finger we want to associate with this one.

First we present some pseudo-code to explain how the finger estimation validation process is done and then we present some examples. In figure 4.13 we have the checks that must be validated to try to change the estimated finger.

If the fret of the note to be evaluated differs from that of the reference note, we are in the first path. Then we go through all the already evaluated notes and check if one of them has the same finger associated if the position between both is different and if both are associated with the same reference note. If these checks are met, we try to change the finger of one of the notes. In figure 4.14 we have the pseudo-code that try to change the finger of one of the notes. If the notes are directly dependent on each other this change has to be made, because we cannot have the same finger associated with two note that have to be heard in one or more instant in the music. If they are not directly dependent, we want to try to change the finger to optimize the

```
if the evaluated note fret != reference note fret:
 for each already evaluated notes n:
  if estimated finger == n associated finger:
  if evaluated note position != n position:
  if reference note associated with the evaluation note
  == reference note associated whith n:
```

Figure 4.13: Checks that must be validated to change the estimated finger.

use of the fingers, but the hypothesis is not impossible if we cannot. If the estimated finger is finger 4, we can only reduce the fingers, since finger 4 is the biggest finger we can use. If the fingers are between 1 and 4, you can either increase or reduce them.

```
if evaluated note and n are dependents:
 if estimated finger == 4:
  we try to decrement the finger of the n note
 elif 1 < estimated finger < 4 :
  we try to decrement the finger of the n note
  if it is not possible:
  we try to increment the finger of the evaluated note
```

Figure 4.14: Change of the same finger associated with two different notes.

Now, let us look at the example we have been following. Here, we have to check if from these notes: [E2:(6,0) finger: 0, B3:(4,9) finger: 1, G3:(3,0) finger: 0, B3:(4,9) finger: 1, G3:(3,0) finger: 0], any of them forbids the use of finger 4 on the note to be evaluated, the E5:(1,12) note. Each note object has a list of the fingers already used. This list lets us know which fingers have already been analyzed by the system. So, in our example the list of E5 already contains finger 4, which is the first finger to be analyzed for this note. We then go through the notes already evaluated and for each note we check if this note has the same finger as the one we want to associate. In our example this does not happen, just see that none of the notes already evaluated has the finger 4 associated. Therefore, this note E5 is associated with finger 4.

We now go to the second note E5 where the already evaluated notes are the following: [E2:(6,0) finger: 0, E5:(1,12) finger: 4, B3:(4,9) finger: 1, G3:(3,0) finger: 0, B3:(4,9) finger: 1, G3:(3,0) finger: 0]. When we evaluate the second E5, the previous E5 note have the same finger that we want to associate with this one. However, besides the condition that both notes have the same finger two other conditions must be met. First, the notes must not have the same position, which is not the case here. Previously it is checked, for each note, if there are dependent notes on the same string. Then we know that these two notes E5 are independent

of each other. If they are independent and played in the same position it makes sense to keep the same finger if possible. The other condition, which is also not met, is that both notes must have the same associated reference note. Notes associated with the same reference note belong to the same set where there are direct and indirect dependencies. Since each reference note is associated with finger 1 and it is through it that we associate the fingers of the other notes, we only consider changing a note finger if there is an impossibility of a note belonging to the same set. The result of this hypothesis would then be: [E2:(6,0) finger: 0, E5:(1,12) finger: 4, B3:(4,9) finger: 1, G3:(3,0) finger: 0, E5:(1,12) finger: 4, B3:(4,9) finger: 1, G3:(3,0) finger: 0].

This example is quite simple since it is not necessary to change any finger. For a more complex example let us look at the following example from figure 4.15, where we have the following notes: [E3: (4,2), C3: (5,3), D4: (2,3), G#4: (1,4)]. This set has only one reference note, which is E3 note. After associating it with finger 1 we go to the remaining ones. The first note to be evaluated is the note C3, where the distance given by equation 4.2 is two. Up until now we only have the reference note evaluated, this note is associated with finger 2. The second note to be evaluated is note D4 where the distance is also 2. In this case, the three conditions mentioned above are met. Finger two is already associated with note C3, both have the same reference note associated with them, and they are not in the same position.

After checking the conditions, it is necessary to see if the notes are directly dependent on each other. If they are it is mandatory that the associated finger is different, because we cannot have the same finger in two different positions. If they are not directly dependent we want to try to change the finger to optimize the use of the fingers, but the hypothesis is not impossible if we cannot. So, we try to change the finger of the note C3 in order to find a possible way to play the hypothesis if the notes are dependent or a more optimal way to play it if they are not dependent. If the finger estimation is finger 4, which is not the case, we can only reduce the fingers, since finger 4 is the biggest finger that we can use. If the fingers are between 1 and 4, you can either increase or reduce them. Then, we try to decrease the finger that went from finger 2 to 1. It is necessary to check if this finger has already been used through the list of used fingers of this note, where for now only finger 2 is found. Finger 1 is added to this list. Without going through the notes already evaluated to see if it is possible to associate this note with finger 1, a set of checks are made to see if this finger is possible to associate. These checks check two things: The first is to verify if the finger that we want to associate is the finger one. If it is, we check if the hypothesis is a barre, if it is not a barre this note cannot be associated with finger 1, if it is a barre, we must check if this note is in the same fret of the reference note, if it is not also cannot be associated with finger 1. The second check is whether the distance between contiguous fingers is large. Above we talked about the maximum distance between the reference note and the note being evaluated. Here we want to find out whether we are associating a finger that is far enough, in fretting terms, from the adjacent fingers, to make it impossible to associate it.

In our example, the attempt to associate finger 1 has failed in the first check, because neither the hypothesis is a barre nor is the note in the same fret as the reference note. After finger 1, it is impossible to decrease the finger any further as there are no more fingers and so the finger of the note that is being evaluated is incremented, since fingers 3 and 4 have not yet been evaluated. The next finger to be evaluated is the finger 3, here it is only checked if the finger has already been evaluated before, and if not, it is associated to the note D4. This is because the association of fingers to notes is made from left to right, from the leftmost fret on the lowest string to the rightmost fret on the highest string. So we know that on the notes already evaluated there are

no notes that can disturb this association.

The next and last note to be evaluated is the note G#4. For now the notes already evaluated are: [E3: (4,2) finger: 1, C3: (5,3) finger: 2, D4: (2,3) finger: 3]. The distance between G#4 and E3 in terms of frets is 3. Since the finger we want to associate is the same as the one associated with the note D4 we try to change it. We try finger 2 for the note D4, however, finger 2 has already been evaluated for this note, so it is decremented to finger 1. Here finger 1 fails for the same reasons as above. As there are no more fingers to decrease, we increment the finger of the note to be evaluated, which is finger 4 where the final association is made.

E4	F4	F#4	G4	G#4	A4	A#4	B4	C5	C#5	D5	D#5	E5	F5	F#5	G5	G#5	A5	A#5	B5
B3	C4	C#4	D4	D#4	E4	F4	F#4	G4	G#4	A4	A#4	B4	C5	C#5	D5	D#5	E5	F5	F#5
G3	G#3	A3	A#3	B3	C4	C#4	D4	D#4	E4	F4	F#4	G4	G#4	A4	A#4	B4	C5	C#5	D5
D3	D#3	E3	F3	F#3	G3	G#3	A3	A#3	B3	C4	C#4	D4	D#4	E4	F4	F#4	G4	G#4	A4
A2	A#2	B2	C3	C#3	D3	D#3	E3	F3	F#3	G3	G#3	A3	A#3	B3	C4	C#4	D4	D#4	E4
E2	F2	F#2	G2	G#2	A2	A#2	B2	C3	C#3	D3	D#3	E3	F3	F#3	G3	G#3	A3	A#3	B3

Figure 4.15: Example Hypothesis

Our last example is more complex, as it requires the use of recursion. Basically, it is sometimes necessary to change fingers of notes that have already been evaluated, and when this is the case, we have to check if this change affects the other notes already evaluated. Our example consists of the following notes: [E3: (4,2), B3: (3,4), E4: (2,5), A4: (1,5)]. The reference note is note E3 which we associate with finger 1. Then we associate note B3 with finger 3 and the note E4 with the finger 4. For now, there is nothing new compared to the previous examples. The distance from the last note, A4, to the reference note is also 4. As in the previous example we decrease the finger of note E4 to 3 and try to match it. This finger has not yet been evaluated for the note in question and passes the checks mentioned above. Then a recursive function is used, because we have several decisions depending on each other. To verify if it is possible to associate finger 3 to note E4 we go through the notes already evaluated, remember that the hypothesis at this moment is like this: [E3: (4,2) finger: 1, B3: (3,4) finger: 3, E4: (2,5) finger: 4/3?].

Note B3 is associated with finger 3. The same checks are made as above, i.e., if there is a note that uses the same finger that we want to associate, if they have the same associated reference note and if they have different positions. Another check is necessary, because now we do not guarantee that the evaluated notes are further to the left with respect to the fret or on the same fret on a lower string. So, we check if note B3 is further to the left than E4, because only then does it make sense to decrease its finger. After all this checking, the finger of note B3 is decremented to two, leaving the hypothesis like this: [E3: (4,2) finger: 1, B3: (3,4) finger: 3/2?, E4: (2,5) finger: 4/3?]. The process of associating note E4 stays in standby in order to see if it is possible to associate note B3 with finger 2. It is possible, because no other note is associated with finger 2, and being this association possible, the association of finger 3 with note E4 is also possible and, therefore, the finger 4 associated with the note A4 is also satisfied.

For the second possible path, in addition to the checks made in the previous path, two more checks are made in order to verify that we constructing a barre.

If we have one note already evaluated that there is not the reference note and the associated finger is finger 1, we know that at least two notes are associated with finger 1, and this only

E4	F4	F#4	G4	G#4	A4	A#4	B4	C5	C#5	D5	D#5	E5	F5	F#5	G5	G#5	A5	A#5	B5
B3	C4	C#4	D4	D#4	E4	F4	F#4	G4	G#4	A4	A#4	B4	C5	C#5	D5	D#5	E5	F5	F#5
G3	G#3	A3	A#3	B3	C4	C#4	D4	D#4	E4	F4	F#4	G4	G#4	A4	A#4	B4	C5	C#5	D5
D3	D#3	E3	F3	F#3	G3	G#3	A3	A#3	B3	C4	C#4	D4	D#4	E4	F4	F#4	G4	G#4	A4
A2	A#2	B2	C3	C#3	D3	D#3	E3	F3	F#3	G3	G#3	A3	A#3	B3	C4	C#4	D4	D#4	E4
E2	F2	F#2	G2	G#2	A2	A#2	B2	C3	C#3	D3	D#3	E3	F3	F#3	G3	G#3	A3	A#3	B3

Figure 4.16: Recursion Example Hypothesis

```
if the evaluated note fret == reference note fret:
 for each already evaluated notes n:
  if n is not the reference note:
  if the finger associated with n is 1:
  if reference note associated with the evaluation note
  == reference note associated whith n:
    estimated finger = 1
```

Figure 4.17: Automatically a barre.

happens when we have a barre. So as the evaluated note is in the same fret as the reference note, the evaluated is associated with finger 1. The pseudo-code is present in figure 4.17.

```
if the evaluated note fret == reference note fret:
 for each already evaluated notes n:
  if n finger == 0:
  if string of reference note > string of n:
  if reference note associated with the evaluation note
  == reference note associated whith n:
    cannot be a barre
```

Figure 4.18: Cannot be a barre.

In figure 4.18 we have a check that excludes the possibility of the hypothesis being a barre. If we have one note already evaluated that is a loose note and the string of this note is below the reference note string and the reference note associated with the evaluated note and the n note is the same, we know that we cannot have a barre, because if this loose string has to be heard during the supposed barre, the position of finger 1 along the fret would not allow this to happen.

For the second possible path, we address a hypothesis that results in a barre. The example is present in figure 4.19. The hypothesis consists of the following notes [F#2: (6,2), B2: (5,2),

F#4: (1,2)]. The reference note is the note F#2 that is associated with finger 1. Next, we try to associate a finger with the note B2. The finger that we want to associate, given the distance, is finger 1. In this case, this note is in the same fret as the reference note. For now, the only note evaluated is the reference note. It is also checked if there are notes with the same finger. In this case, both the note that is being evaluated and the reference one uses the same finger. However, as there is a possibility of a barre in which the finger 1 is used for both notes, we only try to find another finger to be associated if the note that has the same finger is not the reference note. Given this, finger 1 is associated with the note B2 and the hypothesis is assumed to be a barre. Next, we evaluate the note F#4. Now there is a note, B2, which is not the reference note, and it is associated with finger 1 and has the same reference note associated. Because it is also on the same fret as the reference note, it is also associated with finger 1. Because if a barre is being made on that fret, other fingers cannot be used while the barre is being performed.

E4	F4	F#4	G4	G#4	A4	A#4	B4	C5	C#5	D5	D#5	E5	F5	F#5	G5	G#5	A5	A#5	B5
B3	C4	C#4	D4	D#4	E4	F4	F#4	G4	G#4	A4	A#4	B4	C5	C#5	D5	D#5	E5	F5	F#5
G3	G#3	A3	A#3	B3	C4	C#4	D4	D#4	E4	F4	F#4	G4	G#4	A4	A#4	B4	C5	C#5	D5
D3	D#3	E3	F3	F#3	G3	G#3	A3	A#3	B3	C4	C#4	D4	D#4	E4	F4	F#4	G4	G#4	A4
A2	A#2	B2	C3	C#3	D3	D#3	E3	F3	F#3	G3	G#3	A3	A#3	B3	C4	C#4	D4	D#4	E4
E2	F2	F#2	G2	G#2	A2	A#2	B2	C3	C#3	D3	D#3	E3	F3	F#3	G3	G#3	A3	A#3	B3

Figure 4.19: Barre Example Hypothesis

Our last example from this path is a hypothesis that gives the idea of being a barre, but it cannot be. The example is shown in the figure 4.20. The hypothesis consists of the following notes: [B3: (2,0), F#2: (6,2), B2: (5,2), A3: (3,2)]. First the ”0” finger is associated to the loose note, the note B3. Then a finger 1 is assigned to the reference note, the note F#2. Next, we try to associate a finger to the note B2. The distance remains the same as in the previous case, 1. However, now it is not possible to associate the finger 1 to this note, because there is a loose note on one of the strings below the reference note. If this loose string has to be heard during the supposed barre, the position of finger 1 along fret 2 would not allow this to be happen. If the unfretted note were on a string above the reference note string, it would already be possible to make the barre, because the barre starts on the reference note string. As it is not possible to decrease finger 1 of the reference note, the finger of the note to be evaluated is incremented to 2 and associated to the note. Next the note A3 is evaluated, where the distance is 2, given the distance between strings. Here the same process is done as explained in the previous path, we try to decrease the note B2, since it is associated with the same finger, which is not possible, and so the finger of note A3 is incremented to 3.

E4	F4	F#4	G4	G#4	A4	A#4	B4	C5	C#5	D5	D#5	E5	F5	F#5	G5	G#5	A5	A#5	B5
B3	C4	C#4	D4	D#4	E4	F4	F#4	G4	G#4	A4	A#4	B4	C5	C#5	D5	D#5	E5	F5	F#5
G3	G#3	A3	A#3	B3	C4	C#4	D4	D#4	E4	F4	F#4	G4	G#4	A4	A#4	B4	C5	C#5	D5
D3	D#3	E3	F3	F#3	G3	G#3	A3	A#3	B3	C4	C#4	D4	D#4	E4	F4	F#4	G4	G#4	A4
A2	A#2	B2	C3	C#3	D3	D#3	E3	F3	F#3	G3	G#3	A3	A#3	B3	C4	C#4	D4	D#4	E4
E2	F2	F#2	G2	G#2	A2	A#2	B2	C3	C#3	D3	D#3	E3	F3	F#3	G3	G#3	A3	A#3	B3

Figure 4.20: Example of a non-barre

4.5.7 Hypotheses Creation

We mentioned above that the process of generating, checking and creating hypotheses is done incrementally. That is, you take a subset of one or more notes with the same onset belonging to a set of notes dependent on each other, and the various hypotheses are generated and checked as explained above. After the hypotheses are verified, the valid hypotheses are created. For a hypothesis to be valid it is sufficient to check that all notes that belong to the hypothesis have fingers associated with them. If they do not, the hypothesis is not valid and is excluded. The notes of the following subset are added only with the valid hypothesis of the previous subset.

Due to MIDI errors, which quite often create dependencies between notes that do not correspond to the dependencies between notes present in scores of the respective piece of music. It can happen that these dependencies create only hypotheses that are impossible to perform. A simple example is when E2 and G2 are dependent notes in standard tuning. Both only exist on string 6 and, therefore, it is impossible to play them at the same time. In order to always create at least one valid hypothesis for a set of notes, the system splits the set from the note or notes which made all hypotheses invalid. Consider this set: [[E2, E5],[B3],[G2,E3], [G4]]. If notes E2 and G2 are directly dependent on each other, no valid hypotheses are generated given the impossibility of playing both notes at the same instant in the music. Then we split this set into two subsets: [[E2, E5], [B3]], [[G2,E3], [G4]].

4.6 Hypotheses Selection

After all notes in a set have been evaluated and all valid hypotheses have been created, some are excluded. Given the characteristics of the guitar - the frets from the 12th fret onwards already belong to the guitar body, which makes it difficult to use the left hand to press these frets. Given this extra difficulty, the system considers that hypotheses with notes from the 12th fret on are only considered if those notes are notes that only exist from the 12th fret on, or if the exclusion of a hypothesis with notes after the 12th fret empties the set of valid hypotheses. The notes that only exist from the 12th fret on are: E5, F5, F#5, G5, G#5, A5, A#5 and B5. This allows us to cut paths that are not the most optimal.

If in the first set of notes we have these hypotheses: $[h_{11}, h_{12}, h_{13}, h_{14}]$ and in the second set these: $[h_{21}, h_{22}, h_{23}, h_{24}]$, the possible paths correspond to the Cartesian product of these two lists. Each hypothesis has a certain cost and the path which has the lowest cost, i.e., where the sum of the costs of the hypotheses is smallest, is chosen to represent the tablature of the respective piece of music. If we are in the first set of notes, where the first hypotheses are created, only the 10 best hypotheses move on to the next set. It is necessary to make cuts in the number of hypotheses, because the growth in the number of possible ways to play the respective piece of music is exponential. To calculate the cost of each hypothesis, the distance between each fretted note and the remaining fretted notes is calculated. Unfretted notes since they do not use any finger of the left hand do not enter into the distance count. The distance between two notes is given by the equation 4.1.

For the second set of hypotheses, the individual cost of each hypothesis is also calculated as above where the 10 best hypotheses are also chosen. From the 10 best hypotheses in the first set of hypotheses and the 10 best hypotheses in the second set, paths are made. For each path, the distance from one hypothesis to the other is calculated in the same way as the individual

hypotheses are calculated, that is, we calculate the distances of all the notes of one hypothesis with all the notes of the other hypothesis. The distance between sets of hypotheses is only calculated between the current set and the previous one. By calculating the hypotheses individually we are calculating the finger span and by calculating the distance from one hypothesis to the other we are calculating the movement that the left hand has to make.

E4	F4	F#4	G4	G#4	A4	A#4	B4	C5	C#5	D5	D#5	E5	F5	F#5	G5	G#5	A5	A#5	B5
B3	C4	C#4	D4	D#4	E4	F4	F#4	G4	G#4	A4	A#4	B4	C5	C#5	D5	D#5	E5	F5	F#5
G3	G#3	A3	A#3	B3	C4	C#4	D4	D#4	E4	F4	F#4	G4	G#4	A4	A#4	B4	C5	C#5	D5
D3	D#3	E3	F3	F#3	G3	G#3	A3	A#3	B3	C4	C#4	D4	D#4	E4	F4	F#4	G4	G#4	A4
A2	A#2	B2	C3	C#3	D3	D#3	E3	F3	F#3	G3	G#3	A3	A#3	B3	C4	C#4	D4	D#4	E4
E2	F2	F#2	G2	G#2	A2	A#2	B2	C3	C#3	D3	D#3	E3	F3	F#3	G3	G#3	A3	A#3	B3

Figure 4.21: First Path.

E4	F4	F#4	G4	G#4	A4	A#4	B4	C5	C#5	D5	D#5	E5	F5	F#5	G5	G#5	A5	A#5	B5
B3	C4	C#4	D4	D#4	E4	F4	F#4	G4	G#4	A4	A#4	B4	C5	C#5	D5	D#5	E5	F5	F#5
G3	G#3	A3	A#3	B3	C4	C#4	D4	D#4	E4	F4	F#4	G4	G#4	A4	A#4	B4	C5	C#5	D5
D3	D#3	E3	F3	F#3	G3	G#3	A3	A#3	B3	C4	C#4	D4	D#4	E4	F4	F#4	G4	G#4	A4
A2	A#2	B2	C3	C#3	D3	D#3	E3	F3	F#3	G3	G#3	A3	A#3	B3	C4	C#4	D4	D#4	E4
E2	F2	F#2	G2	G#2	A2	A#2	B2	C3	C#3	D3	D#3	E3	F3	F#3	G3	G#3	A3	A#3	B3

Figure 4.22: Second Path.

In figures 4.21 and4.22 we have two alternative paths. The first hypothesis is constituted by the notes in red and the second hypothesis is constituted by the orange notes. To calculate the distance between hypotheses, in this example, we calculate the distance between note A#3 and F#4, A#3 and D4, E3 and F#4, E3 and D4, C3 and f#4 and C3 and D4. The cost of the second path is considerably lower than the cost of the first path, because the notes are much closer together. So, the movement of the left hand along the guitar neck is much smaller. We discussed in section 3.2 that this kind of movement is the most complex one, and is therefore a movement to be avoided.

When there are more than two sets of hypotheses the process is similar to the previous one, however two thresholds are used so that the growth of paths is not impossible to process. For each set the cost of the individual hypotheses is calculated, but in the previous cases the top 10 hypotheses were chosen, From the third set the number of hypotheses can vary as the user chooses, the default threshold is also 10, the user can also make the value vary throughout the evaluation of the music, if the number of different paths is growing exponentially. After the paths are scored the second threshold consists of how far apart in terms of score the other paths are from the top scored path, the higher the threshold the more paths are encompassed, the default threshold is 25, The user can also make this value vary throughout the evaluation of the music.

4.7 Finger Optimization

In figure 4.23 we have the possible paths. One possible path is composed of: $h_{11}, h_{22}, \cdots, h_{m3}$. After the paths are complete, the best path, that is, the path that got the lowest cost is chosen for this phase. In this phase the optimization of the fingers is done. We stated above that all notes have fingers associated with them, taking into account only the context of the hypothesis where the notes are inserted. Here we change the fingers taking into account the previous and the next hypotheses. When we change the fingers of the hypothesis of the second group, we take into account the hypotheses of the first and third groups of hypotheses. The context has great importance in the choice of fingers, because it often happens that the choice of fingers in one hypothesis is not the most optimal, but it facilitates the change to the next hypothesis. From now on, we explain how this whole process is done where we show some examples where changing fingers facilitate the performance of the respective piece of music. Note that since the path has already been chosen, the hypotheses that are spoken of here are the chosen hypothesis of each group.

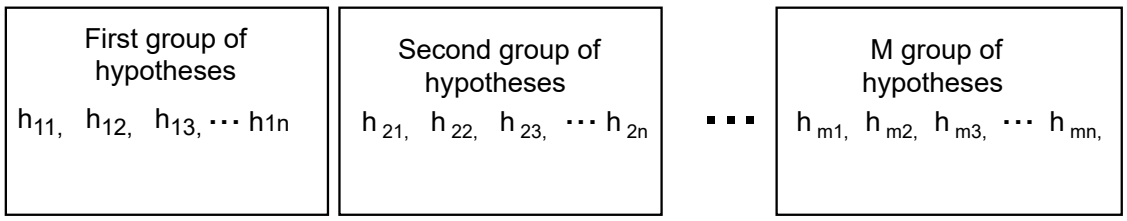

Figure 4.23: Path Construction.

4.7.1 Notes Division By Moments

For each set of notes of the chosen path, a new division is required. This division is similar to the previous division mentioned in section 4.5.5.2 when the existence of a loose note connects the whole set.

However, here a new class is created, the class *MomentDivision*. Each object of this class contains the following attributes: a set of notes, which are the notes that belong to this *MomentDivision* object. The id of the hypothesis that the *MomentDivision* object belongs to and the id of the *MomentDivision* object itself, where the conjunction of the two ids makes this object unique. An attribute consisting of a boolean that indicates whether the fingers of this *MomentDivison* object can be changed or not. A list which indicates which fingers are used by the respective notes of the *MomentDivision* object. A boolean *wasChanged* which indicates if this object has already been changed by the previous or next object, i.e., if given the previous or next context some finger of some note was changed. A list that contains the moments of the longest fretted note in that *MomentDivision* object. Finally, a list indicating by which *MomentDivision* object this one was changed.

From now on the path is not a set of hypotheses, as it was until then, but a set of *MomentDivisions* objects. This new division was necessary because it allows us to divide the hypothesis into smaller parts having more detail in each new set, and thus better understand how the fin-

gers are affected by their external context. If there is not a loose note that links the whole set of notes, the object *MomentDivisions* is composed of all the notes of the hypothesis.

To better understand how this division is made let us look at the following example present in the figure 4.2, where one of the possible hypotheses is as follows: [E2: (6,0), B4: (1,7), B3: (2,0), G3: (3,0), B4: (1,7), B3: (2,0), G3: (3,0), B4: (1,7), B3: (2,0), G3: (3,0)]. Loose notes are separated from non-loose notes. From the fretted notes the divisions are created as explained in section 4.5.5.2, where the fretted notes are divided taking into account the dependency between them. The resulting division of the fretted notes results in three objects *MomentDivisions*: [B4: (1,7)], [B4: (1,7)] and [B4: (1,7)], since all B4 notes are independent on each other and the rest notes are loose notes.

To complete the construction of each *MomentDivisions* object, we need to add the loose notes to the corresponding *MomentDivisions* object. To add the loose notes to existing *MomentDivisions*, we simply check which *MomentDivisions* object they belong to. Each object contains a list of the longest note moments in the set, as discussed above. In our example the list of moments of the first object is as follows: [1,2,3]. By trying to figure out to which *MomentDivision* object the note E2 belongs we check to which object the smallest moment of the note E2 belongs to, or in other words, figure out when the note E2 starts. The E2 note moments are as follows: [1,2,3,4,5,6,7,8,9], because this note is heard throughout the whole set. The E2 note is added to the first object because the first moment of E2 is the moment 1 and the moment 1 appears in the list of moments of the first object. After joining the loose notes the three objects would look like this: [E2: (6,0), B4: (1,7), B3: (2,0), G3: (3,0)], [B4: (1,7), B3: (2,0), G3: (3,0)] and [B4: (1,7), B3: (2,0), G3: (3,0)].

Suppose that in addition to the previous notes we have three new ones: [E4: (1,0), B3: (2,0), G3: (3,0)] where their moments are: [[10,11,12], [11], [12]] respectively and that the moments of note E2 were extended to [1,2,3,4,5,6,7,8,9,10,11,12]. These three notes are independent of the others except note E2. Therefore, they do not belong to any of the previous moments, but to a new moment. A special moment which is made up of loose notes only.

4.7.2 Changeable Fingers

After dividing the hypothesis, we check whether the fingers of each *MomentDivisions* object can be changed. This information allows us to see if there are moments when the notes are already with the final finger choice and therefore no other possibilities are tried. The *MomentDivisions* object when only one finger is being used, either because it has only one note, or because the others notes are loose notes, can be changed. Since only one finger is being used, three other fingers are liable to be used. When two fingers are currently being used, only the use of fingers 1 and 4 can make it impossible to change fingers. If the distance between the notes in terms of their frets using the respective fingers is greater than or equal to 4 we know that the use of other fingers is impossible, this is because the distance forces the use of the two fingers on the ends. All the *MomentDivisions* objects when 4 fingers are being used cannot be changed, this is because the way we associate the fingers when constructing the hypothesis is optimized and the most correct without taking the context into account, however when 4 fingers are being used the rules used previously for finger association override a possible optimization taking the context into account. *MomentDivisions* object when 3 fingers are being used can be changed.

4.8 Change Fingers

Here we explain how the fingers are changed when appropriate. For easier reading each *MomentDivisions* object is called mom.

In figure 4.24 are present the moms resulting from the division explained in the previous section. We start with mom 1 and we analyze the mom 1 and the next mom. The analysis that is done is to understand if we could change any finger of either mom 1 or mom 2 in order to facilitate the passage from mom 1 to mom 2. Then we go to the mom 2 where we analyze mom 2 with the previous and next moms and so on. However, it is necessary to go in the opposite direction when any of the fingers of a mom is changed. Suppose that we are in mom 4 and when we analyze the three moms: mom 3, mom 4 and mom 5, one of the fingers of mom 3 is changed. We need to check if this change alters any of the fingers of the previous moms. However, we do not have to go through all the existing moms before mom 3, if, for example, mom 2 were unchanged by the new mom 3 change, there would be no need to check mom 1.

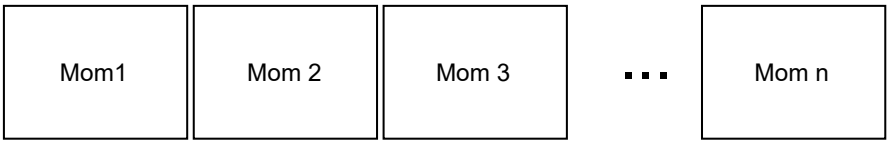

Figure 4.24: Moms.

In the analysis of the moms two different approaches are followed: the first one is followed when we are dealing only with *MomentDivisions* objects with only one fretted note. the second one is followed when we are dealing with at least one *MomentDivisions* object with more than one fretted note. However, for the system to understand what kind of moments it is analyzing it needs to make some checks that are covered next.

4.8.1 Checking the Moms

When the system is analyzing the moms the first check to be made is whether the moms are changeable. If no mom that is being analyzed is changeable then the system goes to the next mom. When only one mom is changeable, the analysis is done knowing that only the mom that is changeable can be changed, i.e., that is, it is possible that the changeable mom can be changed in order to make it easier to go from one mom to another. When all moms are changeable, the analysis is done knowing that all moms can be changed. To check if a mom is changeable just check the attribute associated with each object that gives us this information. The second check checks if at least one mom has more than one fretted note. To check this we check if the list which indicates which fingers are used by the respective notes of the mom has more than one finger. In the first approach we are dealing with moms that contain only one note and there is one case that let us know that no changes are made. If the fretted note of all the moms being analyzed use the same finger and all the notes are in the same position, then no change is made. If the notes are in the same position, it makes sense that they use the same finger, since then we just keep the finger where it already is.

4.8.2 Moms with Only One Fretted Note

We show now an example of the first approach, that consists of the first two bars of the piece Spanish Romance 4.25. As the issue of creating *MomentDivisions* objects has already been discussed, we only present them next: Mom1: [E2: (6,0), B4: (1,7), B3: (2,0), G3: (3,0)], Mom2: [B4: (1,7), B3: (2,0), G3: (3,0)], Mom3: [B4: (1,7), B3: (2,0), G3: (3,0)], Mom4: [E2: (6,0), B4: (1,7), B3: (2,0), G3: (3,0)], Mom5: [A4: (1,5), B3: (2,0), G3: (3,0)] and Mom6: [G4: (1,3), B3: (2,0), G3: (3,0)]. We start by evaluating the first mom, [E2: (6.0), B4: (1.7), B3: (2.0), G3: (3.0)] and we look only at the next mom, since there is no previous mom. We know that both moms are liable to be changed. Since only one finger is used at each mom, we also know that both moms use the finger 1 and since both notes have the same position no changes are made. The next mom to be evaluated is mom 2, [B4: (1,7), B3: (2,0), G3: (3,0)], which now is evaluated with the previous mom and with the next mom. The three moms are changeable, all use only one finger, finger 1, and finger 1 is used in the same position at all three moms. So, no changes are made. The same process happens when evaluating mom 3.

In evaluating mom 4 we analyze moms 3, 4 and 5. All three moms are changeable, they only use one finger, finger 1. However, the note that uses finger 1 at mom 5 is in a different position than the notes of moms 4 and 3. With 4 possible fingers to use, it does not make sense to use the same finger for different positions and so we try to change some finger of one of the 3 notes. The three notes are: [B4 (1,7) f: 1, B4 (1,7) f: 1, A4 (1,5) f: 1]. We know that the first two notes are in the same position, so we just need to check which of the notes in moms 4 and 5 has the highest fret. In this case it is note B4, so we calculate the new finger to this note. The calculation of the new finger is given by the following equation:

$$FingerCalculus = abs(fret_i - fret_{i+1}) + finger_{i+1} \tag{4.3}$$

Where $fret_i$ is the fret of the B4 note, $fret_{i+1}$ is the fret of the A4 note and the $finger_{i+1}$ is the finger associated with the A4 note. The result of this equation is: $abs(7-5)+1) = 3$.

The new finger is the minimum between this result and 4. The finger that is changed when comparing two notes is the finger of the note with the highest fret. The finger of the note with the lowest fret is only changed when the distance between two adjacent fingers is large. So, the new finger of note B4 is updated along with the following updates: The list of used fingers of mom 4 is updated, because the new used finger is 3 and not 1, the flag *wasChanged* mentioned about above becomes true, because this moment was changed and it is added to mom 4 the information that it was mom 5 the reason for the change.

Since mom 4 was changed we need to see if this change alters mom 3. Since they have the same position, the same change is made for mom 3. Then the *wasChanged* attribute of mom 3 is checked, and if it is true, it is necessary to check the previous moms. Because it is necessary to understand if the change of mom 3 does not influence any change of mom 2. So, next we evaluate the moms 3 and 2. Both are changeable, both use only one finger only this time they use different fingers, mom 3 uses finger 3 in position (1,7) while mom 2 uses finger 1 in the same position. As in both moms the note has the same position, something needs to be clarified, which note has priority over the other? To understand which of the moms has priority over the other it is necessary to verify which of the two moments was the last to be modified. In this case only mom 3 was modified, thus having priority over mom 2. Then we try to change mom 2, where the process is the same as before, the *Fingercalculus* is 3 and therefore the note

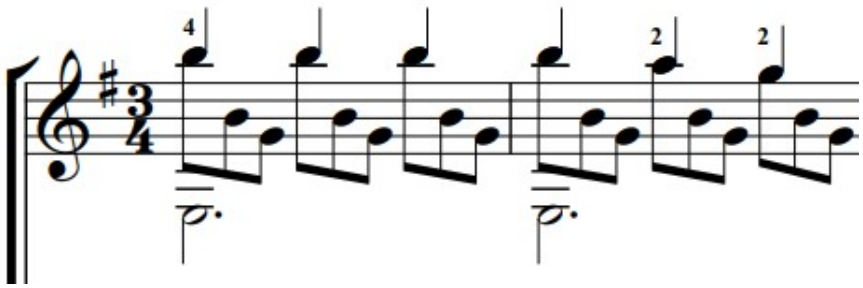

Figure 4.25: The first two bars of Spanish Romance

of mom 2 is associated with finger 3, the same finger that is associated with the note of mom 3, which makes since the note is in the same position. As the mom 2 has been changed it is necessary to check mom 1, where the process is the same since it is also the same note in the same position with different fingers being associated.

We now evaluate mom 5 where we analyze moms 4, 5 and 6. All three moms are changeable, they use only one finger, however, different fingers. Mom 4 uses finger 3 for position (1,7), mom 5 finger 1 for position (1,5) and mom 6 finger 1 for position (1,3). In this case, unlike the previous cases, all positions differ from each other. Then we need to check the order of the frets, in this case we are in a descending movement. So, mom 6 has predominance over mom 5 and mom 5 over mom 4. In other words, the smallest fret always has predominance over the highest fret, if the fret is the same, the highest string has predominance over the smallest.

We try to change the finger of the note of mom 5 by looking at mom 6. The *FingerCalculus* is 3 being this the finger associated to the note of mom 5. Then we change the finger of mom 4 taking into account the new update of mom 5. The distance between the frets is 2 and the *Fingercalculus* is 5, so the new finger that we associate to the note of mom 4 is the finger 4. We mentioned above that the finger of the smallest fret is only changed when the distance between two adjacent fingers is large. The note of mom 5 is associated with finger 3 while the note of mom 4 is associated with finger 4. The distance between the two frets is two, which implies a large stretch between finger 3 and finger 4, then the finger of the note with the lowest fret is decremented by 1. As mom 4 was changed it is necessary to verify how this change influences the previous moments as explained above. The end result, after optimizing the fingers is this: M1: [E2: (6,0) f:0, B4: (1,7) f:4, B3: (2,0) f:0, G3: (3,0) f:0], M2: [B4: (1,7) f:4, B3: (2,0) f:0, G3: (3,0) f:0], M3: [B4: (1,7) f:4, B3: (2,0) f:0, G3: (3,0) f:0], M4: [E2: (6,0) f:0, B4: (1,7) f:4, B3: (2,0) f:0, G3: (3,0) f:0], M5: [A4: (1,5) f:2, B3: (2,0) f:0, G3: (3,0) f:0] and M6: [G4: (1,3) f:1, B3: (2,0) f:0, G3: (3,0) f:0]. We went from a set of moms where all non-fretted notes were associated with finger 1, since they were reference notes of the mom itself, to a much more optimized association that takes into account the context where the note is inserted.

4.8.3 Moms with More than One Fretted Note

We now look at the second approach where at least one of the moms contains more than one fretted note. In the first approach where all moms contain only a single fretted note, the change of finger associated with that note has no implications for the other notes of the moment, because either the moment contains only that note or because it contains only loose notes besides

that note. Loose notes are played without the need to press any fret, so it is not necessary to use any finger of the left hand to play them. If we are optimizing the use of the fingers of the left hand, the loose notes are not taken into account in this optimization. Since moments that contain only one fretted note only depend, in terms of finger optimization, on the note in question, we are allowed to approach these types of moments as we showed above, by checking the arrangement of frets between notes of the various moments.

Pretend that we have the following mom: M1: [A3: (2,3) f: 1, D4: (3,2) f:2] where the notes A3 and D4 are dependent on each other, i.e., they have to be heard during a certain period of time in the music. Suppose that we have a mom M2 that causes a change from finger 1 to finger 2 of note A3. If we did not consider the other notes of the mom 1, we would have the following mom1: [A3: (2,3) f: 2, D4: (3,2) f:2]. This mom would be impossible to perform, because if notes A3 and D4 are dependent, they cannot both be associated with finger 2, because the same finger cannot be on two notes at the same time. In fact, if the finger associated with note A3 was finger 2, the finger associated with note D4 that would make the most sense would be finger 3. This shows us that a change in one finger of a note of a mom with more than one fretted note, can lead to changes in the other notes of the mom. So, it is necessary to make sure that the change of a finger from a note is made taking into account the possible changes propagating from that change.

This approach is divided into three parts: the first part consists in comparing common positions between moms, the second part consists in comparing the last notes of mom i with the first notes of mom i + 1 and the third part consists in checking if the change of a mom triggers changes in the previous moms, i.e., if there is a change at mom i - 1 it is necessary to check if this change influences any of the previous moms. These three parts are explained in detail below, but the general idea is, given two moms i and i + 1, to understand whether by changing any of the fingers of one or both moments we can facilitate the passage from mom i to mom i + 1.

4.8.3.1 Common Positions Between Moms

The first part of this second approach consists in checking whether there are positions in common between two moms. If there is a position in common between the two moms where each position is associated with a different finger, it is possible that by changing the finger of one of the moms, the transition from mom i to mom i + 1 is easier to perform. We want to try that for common positions the finger that was used at mom i is also used at mom i + 1. Let us consider that moms 1 and 2 are as follows: mom1: [E3: (4,2) f: 1, D4#: (2,4) f:3], mom2: [A3#: (3,3) f:1, D4#: (2,4) f:2]. In this example the note D4# is a common position and is associated whit finger 3 in mom 1 and associates with finger 2 in mom 2. We want to create a better mom1mom2 combination than the current one. To do this we create two alternative moms which are copies of the mom 1 and mom 2 respectively. These copies have a small difference which is the finger associated with the note in common between moms. So, alternative mom 1 would be: mom1 alternative: [E3: (4,2) f: 1, D4#: (2,4) f:2] and alternative mom 2 would be: mom2 alternative: [A3#: (3,3) f:1, D4#: (2,4) f:3]. The fingers associated with the D4# in the alternative moms are switched, the finger that was associated with the D4# note of mom 1 became associated with the D4# note of mom 2 and vice versa.

As we said above, when a finger on one of the notes is changed it is necessary to check if this change alters anything within the mom itself. To check if any changes are needed we do

the same as in section 4.5.5 where the hypotheses are checked. The process is the same with a few minor changes. In section 4.5.5 we described the concept of reference note, which was the leftmost note on the guitar neck and if there were two or more notes in the leftmost fret on the guitar neck, it was the note that was on the lowest string. Here, the reference note is the leftmost note on the guitar neck, or if there are two or more notes in the leftmost fret on the neck, it is the one on the lowest string among the notes in common between the moms. The reason is that here we want to keep the fingers of the common notes that were assigned when creating the alternative moms, and from that assignment associate the fingers with the notes that are not common between the moms. Looking at the mom 1 alternative, we have that the note D4# is associated with finger 2 and it remains for us to evaluate the note E3. Here the calculation of the distance is different from the one explained in section 4.5.5. Before, since the reference note was the leftmost note on the guitar neck, we knew that all the other notes to be evaluated would be further to the right on the guitar neck, thus, for that notes, higher fingers than the finger of the reference note are always chosen. Here there is no guarantee that the evaluated note is not further to the left than the reference note. If the note is further to the left than the reference note, the possible finger is given by the following equation:

$$\hat{FingerCalculus} = finger_{refnote} - abs(fret_{evalnote} - fret_{refnote}) \tag{4.4}$$

If the note is further to the right on the guitar neck the possible finger is given by the following equation:

$$\overline{FingerCalculus} = finger_{refnote} + abs(fret_{evalnote} - fret_{refnote}) \tag{4.5}$$

If the note to be evaluated is in the same fret the possible finger is equal to the finger value of the reference note. In our example the possible finger is 0: 2 - abs(4-2), where the first two is the finger value of the reference note, the number for is the fret value of the reference note and the second two is the fret value of the evaluated note. We are trying to associate a fretted note with the finger "0", so this moment becomes impossible to play and is discarded.

Then the same process is done for the mom2 alternative: [A3#: (3,3) f:1, D4#: (2,4) f:3]. First the note D4# is associated with finger 3, then we evaluate the note A3# in order to see if the new finger associated with the note D4# change the finger of this note. The finger calculus is: 3 - 1 = 2. Then we do the same process as in section 4.5.5, where we check whether the finger we want to associate is possible to be associated. In this case, it is possible, and the second alternative mom results in the following: mom2 alternative: [A3#: (3,3) f:2, D4#: (2,4) f:3]. After analyzing the alternative moms, we are left with three possible moms: mom 1, mom 2 and alternative mom 2. We then have to figure out which is the best combination, if the best combination is the one we had before: mom 1 and mom 2, or if the best combination results in the substitution of mom 2 by alternative mom 2.

If besides the note D4#, the note E3 was also common to both moms and was associated with different fingers at both moms, the alternative moms created would no longer be two as in the previous example. the first alternative moms with the fingers associated with the note D#4 switched, the second with the fingers associated with the note E3 switched, and the third with the fingers associated with the note D#4 and the fingers associated with the note E3 switched.

4.8.3.2 Choice of the best combination

Consider the example of the previous section where we have two possible combinations. still without taking into account the behavior of the system in choosing the best combination, let us think about what the best combination would be. Looking at the first combination: mom1: [E3: (4,2) f: 1, D4#: (2,4) f:3], mom2: [A3#: (3,3) f:1, D4#: (2,4) f:2], let us think about the effort needed to go from mom 1 to mom 2. Looking at finger 1, it is used in both moms. Of course, the difficulty of this transition, the transition of finger 1 from position (4,2) to position (3,3), depends on the speed of the music, on the temporal restriction imposed. But imagined that the music is played at a high speed this transition would bring a high degree of difficulty. On the other hand we have a common note that is being played with different fingers which implies a change of finger in order to play the same note, which in fact in terms of effort, is a useless effort. Of course, there may be times when finger changes to play the same note are necessary and even quite useful, but this is not the case.

Let us now look at the second combination: mom1: [E3: (4,2) f: 1, D4#: (2,4) f:3], mom2 alternative: [A3#: (3,3) f:2, D4#: (2,4) f:3]. In this second combination, finger 1 is no longer used at both moms, where it has been replaced by finger 2 at the second alternative mom. Therefore, instead of the transition that finger 1 had to make from one mom to the next, a finger is now used that is free and ready to be used, thus causing no effort. As for the note in common between the two moms, in this second combination both use the same finger, just keeping the finger where it already is from one mom to the next. The effort required to move from one mom to another in the first combination is much higher than the effort required in the second combination, thus showing that an optimization is made in the choice of fingers taking into account the context of the music.

The system has to assign some sort of score to each combination in order to be able to evaluate which of the combinations is the best combination. Two heuristics are made for each combination: the first heuristic is the cost of common fingers and the second heuristic is the cost of common positions.

The first heuristic checks the fingers in common between the two moms, and if the commons fingers are in different positions the cost is as follows:

$$CommonFingersCost = abs(fret_{note1} - fret_{note2}) + abs(string_{note1} - string_{note2})/2 \quad (4.6)$$

The distance between strings is divided by two, because the distance between strings does not have as much weight as the distance between frets. The second heuristic checks if there are positions in common, if the common positions have different associated fingers the cost is as follows:

$$CommonPositionsCost = abs(finger_{note1} - finger_{note2}) \quad (4.7)$$

The greater the distance between the fingers the longer the path to press the same position with another finger. The final cost of the combination is the sum of the cost of these two heuristics, and the combination with the lowest cost is chosen.

4.8.3.3 Comparing Notes Between Moms

If there are no common positions between the two moms, we follow to the second part of the approach. If we think about the set of notes between the two moms, the crucial notes when we pass from one mom to the other are the last notes to be heard at mom i and the first notes to be heard at mom i + 1. Because they are the last notes that have to be pressed in mom i, and the first notes that have to be pressed in mom i + 1. Given this, the last notes to be heard in mom i and the first notes to be heard in mom i + 1 are extracted. Then it is checked whether all the notes extracted are loose notes at any of the moms. If they are, no changes are made in moms. If we think that the last notes of mom i are loose notes, i.e., it is not necessary to press any fret with the left hand for the notes to be heard, when we move to the first notes of mom i + 1, all fingers are free and ready to be used, since no optimization of the choice of fingers is needed. If the first notes to be heard from mom i + 1 are loose notes, it means that the passage from mom i to mom i + 1 was done without problem, because no finger of the left hand is needed to hear the first notes of mom i + 1.

If there is at least one fretted note in each set of notes that has been extracted from the moms, it is possible that there is some change in the fingers that could facilitate the passage from mom i to mom i + 1. Let us imagine that the notes extracted from each mom are as follows: notes extracted from mom1: [E3: (4,2) f: 1, C3: (5,3) f:2, D4: (3,4) f: 3] and from mom2: [C4: (3,5) f: 1, B4: (1,7) f: 3]. The goal is to figure out what fingers exist in common between the two sets. Between these two sets there are two fingers in common, finger 1 and finger 3. So we try to find alternatives to these fingers in order to understand if by changing these fingers the passage from mom 1 to mom 2 is facilitated. A number of different possibilities are created that then result in the creation of alternative moms as in the case discussed in section 4.8.3.1. Looking at the notes that are associated with finger 1 it is necessary to check if they are in different positions, if they were in the same position, we would not be interested in trying to change the associated fingers, since they are not we try to check which possible alternatives fingers are useful to try.

To do this we check the arrangement of the frets between the positions with common fingers. Since the common finger is finger 1, we know that we cannot decrease its value, since it is the leftmost finger. So, checking the frets we see that the rightmost note on the guitar neck is the fret of note C4: (3,5). So one of the possible changes would be to increase the value of the finger from note C4 to finger 2. Let us save these possible changes. Possible changes for the mom1: [], for the mom2: [(C4, 2)], where the first position of the tuple is the note of the finger we want to change and the second position is the new finger we want to associate. We now move on to finger 3, where we also check if the positions associated with finger 3 are equal. As they are not, we check the arrangement of the frets between the two notes. As the common finger is finger 3 we know that we can either decrease or increase its value. For the leftmost note on the guitar arm the finger value is decreased to two and for the leftmost note on the guitar arm the finger value is increased to 4. Thus the possible changes are: mom1: [(D4, 2)] and mom2: [(C4, 2), (B4, 4)].

Our goal is to reduce the fingers in common between these two sets, so we do not want to change a finger from a note to another finger that is already at the other mom, that is, if we change the finger of note C4 from finger 1 to finger 2, instead of there being a note at each moment with finger 1 there are a note at each mom with finger 2, since at mom 1 the note C3 is associated with finger 2.

Then we add the concept of danger finger. This concept consists of checking whether one of the new possibilities we want to try out makes new fingers common between moms. To do this, we check if any of the fingers here: mom1: [(D4, 2)] exists in the notes extracted from mom 2 and if any of these fingers: mom2: [(C4, 2), (B4, 4)] exists in the notes extracted from mom 1. In our example only the attempt to associate finger 2 with the note C4 can make finger 2 common to both moments. So we compare the frets between notes C4 and note C3 which has finger 2 associated at mom 1. If the fret of note C4 is greater than the fret of note C3 the value of the finger is increased to 3, if it is less the value of the finger becomes 1. In our example, the fret of C4 is greater than C3, so a new possibility is added to the possibilities of mom2: [(C4, 2), (B4, 4), (C4, 3)]. The possibility of adding finger 2 to the note C4 is not excluded because even making a new finger common between the moms can facilitate the passage between the mom1 and mom2.

The creation of the alternative moms happens in the same way as in the part one. For the mom1 only one alternative mom is created since we have only one different possibility, which is to associate finger 2 to the note D4. For the mom2 5 alternative moms are created, three are considering the new possibilities individually and 2 are considering the new possibilities given the Cartesian product between them. The new possibilities of mom2 can be seen like this: [(C4, 2), (C4, 3)], [(B4, 4)]. So the resulting Cartesian product is as follows: [(C4, 2), (B4, 4)], [(C4, 3), (B4, 4)]. After the alternative moms are created, they are processed in the same way as in the part 1 of this approach, which results in various mom1mom2 combinations. The combination with the lowest cost is chosen.

4.8.3.4 Changes propagated to previous moments

The third part of this approach checks if a change in mom propagates changes to the previous moms. When we are dealing with the evaluation of the first mom, where we look at the first and second moment, if there is a change at mom 1 or mom 2, that change not have consequences for the previous moms since this is the first and second moms of the music. It is only in the following moms that a change in one mom can trigger changes in the previous moms.

When we consider 3 moms, what happens from mom 2 on, our approach is as follows: we analyze the moms i and i + 1, remember that moments i - 1 and i have already been evaluated in the previous iteration. If moms i and i + 1 have positions in common we follow the first part of our approach, if not we follow the second part of our approach. If the mom i was changed, it means that when we calculate the costs of the combinations between mom i and mom i + 1, the chosen combination has one of the alternative moms of mom i. Therefore it is necessary to check whether this change propagates changes in mom i - 1. If the mom i in the previous iteration was changed by mom i - 1, it means that the mom i was changed by both mom i - 1 and mom i + 1. When a mom i is changed by both the previous and the next mom, the three moms are considered and the combinations become the combinations between moms i - 1, i and i + 1. These combinations are created with the same process: it is checked if there are positions in common between mom i - 1 and i and if there are positions in common between mom i and i + 1. If there are, the first part of the approach is followed, if not , the second part is followed. The combination with the lowest cost is chosen. If mom i - 1 is changed, i.e. if the chosen combination contains an alternative mom of mom i - 1, it is necessary to check if mom i - 1 has also been changed by mom i - 2. If it was, it means that mom i - 1 was changed by both mom i and mom i - 2, so the combinations of the four moms are considered. When the mom with the smallest index is unchanged, the back propagation ends and we move on to the

next iteration. If mom i is changed by mom i + 1, but was not changed by mom i - 1, we only consider the combinations of mom i - 1 and mom i.

5 Evaluation

In this section we describe how we evaluate our system. After the system receiving as input a MIDI file and returned, as output, its score, we evaluated the chosen positions of each note and their associated fingers. While the first is evaluated from two different angles where, on the one hand, the positions given by the system are compared to the ground truth positions and, on the other hand, three guitar players evaluated a smaller set of tunes to see if the chosen positions which are different from the ground truth positions are also valid. We tried to evaluate if, even when the system chooses positions that are different from the ground truth positions, the chosen positions are also good alternatives to the ground truth positions. Since the choice of fingers is much more subjective and there is no ground truth, the evaluation of the associated fingers is done with three guitar players who evaluate, for a smaller set of pieces, if the fingers chosen by the system are the best choice or one of the best choices, since the choice of fingers can vary from guitar player to guitar player.

5.1 DataSet

The dataset used to test the system consists of 16 pieces of classical music. Table 5.1 provides information about the 16 pieces, about their period and their composer. The period ranges from the Renaissance period that occurred between 1400 and 1600 until the 20th century.

Table 5.1: Dataset.

Dataset		
Composer	**Pieces**	**Period**
John Dowland	John Smith	Renaissence
Sebastian Bach	Allemande 996	Baroque
Leopold Weiss	Fantasie	Baroque
Francisco Tarrega	Adelita	Romantic
Napoléon Coste	Barcarole	Romantic
Luigi Legnani	Caprice op20 n2 Caprice op20 n3 Caprice op20 n9 Caprice op20 n15	Romantic Romantic Romantic Romantic
Matteo Carcassi	Op5 n6	Romantic
Isaac Albeniz	Asturias	Post-Romantic
Augustín Barrios	La Catedral Prelude in C Minor Estudio del Ligado Re	Folk Folk Folk
Enrique Granados	Dedicatoria	20th Century

In order to understand the robustness of the system, we have tried to diversify the set of pieces to be evaluated. Having music from several periods and several composers, we have a greater diversity of musical characteristics proper to each of the different periods and the different composers. We also try to evaluate the system with pieces, which in the context of classical music, are very well known and important. Every classical guitar player studies, throughout his or her life, pieces such as the Asturias and the Cathedral. They are therefore important pieces in the context of our domain, classical guitar music.

The tablatures for the dataset, that is, our ground truth were made by Allen Mathews who has a website[1] where he shares a large set of sheet music and tablatures for free. Allen Mathews has a B.A in classical guitar performance and is a classical guitar teacher and concert performer having also released a classical guitar album in 2010. Since the tablatures created by Allen Mathews are in pdf format, as in figure 5.2, the tablature information needs to be converted into txt format manually.

```
6 0
3 0
2 0
3 0
1 0
2 0
1 3
2 5
1 7
2 8
1 8
3 9
6 2
4 4
3 2
4 4
2 4
3 2
1 2
2 4
```

Figure 5.1: Fantasie txt example

Initially we aimed to evaluate 50 different pieces, but given the difficulty in evaluating each piece, the number was reduced to 16. The work of converting the information from the tablature to txt was done for the 50 musics, where more than 30,000 notes were converted into txt format. For each piece a txt file was created where the position of each note was taken from the tablature and written to the txt file. In figure 5.1 we can see the beginning of the tablature extraction presented in figure 5.2 that corresponds to the piece *Fantasie* by *Weiss*. The first note is on string 6 fret 0, the second is on the third string fret 0, and so on. After creating this txt, a script was used to transform the information equal to the system's output. The script transforms the first line of this txt where only the following information is present: 6 0 into: Name: E2 position: (6, 0) id: 1, where two new pieces of information are added, the name of the note and its id.

[1] https://www.classicalguitarshed.com/

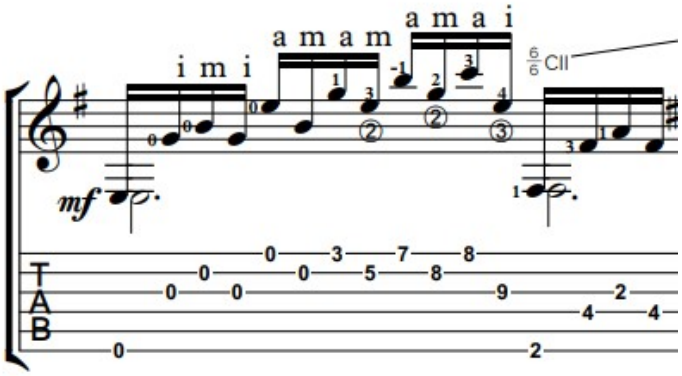

Figure 5.2: Fantasie - Weiss

Besides creating the Dataset, we have the MIDI files that are our system input and a set of txt files that are our ground truth. The idea was to automatically compare the system output with its ground truth, but that was not possible, because the MIDI files contain many errors.In fact there are so many errors that we divided them in three types of errors. The first kind consists of wrong notes which are divided into wrong notes that are replacing ground truth notes or wrong notes that are extra to the ground truth. We take as a piece of ground truth: [A4: (1,5), B4(1,7)] and in MIDI this set of notes has been extracted as: [D3: (4,0), A4#: (1,6), B4(1,7)]. If the notes A4 and A4# have the same behavior, i.e., if they have the same duration, if their dependency on the note B4 is the same, the note A4# is considered a wrong note replacing the ground truth note. However note D3, having no ground truth correspondent, is a wrong note which is extra, since it has no ground truth correspondent. Obviously, when there are wrong notes in the MIDI, the system cannot find the correct positions, which are in the ground truth, since they are not the same note. Since the choice of positions for each note takes into account the context where the note fits in the musical piece, the existence of a wrong note can have a negative impact on the choice of positions for the remaining notes.

The second error is the lack of notes present in the ground truth. In this case suppose the MIDI file does not contain the note information and therefore, when extracting the notes from MIDI the system has no way to extract the notes that exist in the music. Finally, the third type of error and the most difficult to figure out is due to dependency errors between notes. This third type are notes that in the ground truth are independent on each other but in the MIDI file are dependent and vice versa. This error is quite visible in a case that was quite common throughout the evaluation of the notes. In the music *Asturias*, one of the pieces evaluated, it is quite common for the note B3 to be played twice in a row. In the ground truth these two notes are independent and played at the same position, position (2,0). However, the corresponding MIDI considers the two notes directly dependent on each other, which makes it impossible to use the same position for both notes. As we saw above the way notes depend on each other and notes independent of each other are treated is quite different, so this error means that the choice of positions can be quite different from the ground truth positions, without any fault on

the part of the system. In 4.5.7 section we talked about how errors of this type forced the system to split the dependent note sets, because sometimes MIDI creates dependencies that make the note set impossible to play.

A subset of three MIDI files was corrected and it was found that on average 20% of the MIDI files contain errors of these 3 types, with type 3 being the most common error. This is not to say that if these errors did not exist, the system would always find the correct position of the notes, but we know that in these 20% of errors the system has much more tendency to make mistakes in the choice of positions because it is given wrong input. In the first and second type of errors the implication is direct. In the first type, as the notes are wrong, the choice of positions are logically wrong, and in the second type of errors, that is, the lack of notes that exist in the ground truth also logically imply that the note in the ground truth is not identified. However, the third type does not guarantee that the system makes a mistake in the choice of the position, because even though the dependencies are different the system can choose the correct positions when possible.

Since there is a large percentage of errors in MIDI files, files evaluation had to be done manually. Therefore, given the output the system checks, looking at the ground truth, which positions are correct and which are wrong, which notes are missing and which notes are extra notes present on the MIDI. Given the need to evaluate manually the output of each piece of music (which for some pieces was up to 3 days of compilation) it was not possible to evaluate the 50 pieces of music that we had initially planned to evaluate.

In appendix II there is an example of the notation of a midi file, so that you can see how difficult it is to change it and correct existing errors.

5.2 Note Positions Evaluation

In this section we discuss the evaluation of the system's choice of note positions. As mentioned above, for the evaluation of the positions, the output of the system is compared with the ground truth of the musical pieces and they were also evaluated by three guitarists.

5.2.1 Evaluation By Guitar players

In the evaluation of the system by users, three guitarists evaluated a set of 5 musical pieces from the dataset to understand the quality of the alternative choices, in relation to the ground truth ones. One of the guitar player evaluators had classical guitar lessons with a teacher from the age of 12 to 18, attended the Hot Club studying jazz guitar and has a 5th degree from the Metropolitana de Lisboa Conservatory. He also accompanies a choir he founded and is a founding member of the ISCTE orchestra. Another evaluator studied jazz guitar at the *Hot Club* while the third evaluator has a degree in basic education teaching at the *Instituto Politécnico de Lisboa*. He also has a classical guitar course at the *Academia de Amadores de Musica de Lisboa*. He has been teaching for over 20 years, is the founder and composer of the band *Café D'Alma* that has an album released in 2016 among many other musical projects.

When evaluating this set of 5 pieces, it was unanimous that the system always finds playable alternatives. This is not to say that they are better alternatives than the ground truth ones, but that given the context where the positions are inserted, they are easily playable, not

creating extreme difficulties for the guitarist who plays them. When we say that a set of notes (positions) are playable, we mean that, first, notes dependent on each other can be played at the same time. For example, if the system assigned two dependent notes to the same string, that set of notes would not be possible to play, since dependent notes cannot be played on the same string. Second, that dependent notes are not at a very large distance from each other, since if they were it would not be possible, given the physical issue of the fingers of the left hand, to perform them. This is what we wanted to ensure in the first place, that the system always chooses playable positions. We do not want the system to create impossible tablatures to play. The tablatures created by the system may not be as good as the tablatures created by professional guitarists, but since they are always playable, they have value.

Now we need to understand how good the alternatives created by the system are. The heuristics that the system uses to choose the best positions take into account the finger span between notes and the distance between notes along the guitar neck, as discussed in section 4.6. However, there are pieces that use a regular pattern of the right hand, such as the *Spanish Romance*, which throughout the piece plays the strings 1,2,3,1,2,3. This entails that sometimes sets of positions chosen are more difficult to perform but keep that right hand pattern constant. Our system does not take into account the behavior of the right hand and therefore does not take into account these patterns embedded in pieces of music. We talked in section 3.2 about two other factors in the choice of note positions, the musical and cognitive factors. These types of factors can override the biomechanical factor, i.e., instead of looking for the easier set of positions, another less easy set is looked for by making use of the other two factors. Our system only takes into account the biomechanical factor, always looking for the easiest set of positions to execute. Another reason why there are differences between the systems choices and the ground truth is due to the errors that exist in MIDI, especially the errors in the dependencies between notes. The choice of positions depends heavily on the dependencies between notes as we have seen throughout the 4 section. A change in the dependencies between notes can completely change their context, forcing the chosen positions to be different from the ground truth.

In conclusion, the system chooses alternative possible positions to play, sometimes it chooses alternatives that are easier to play than the canonical ones but that were chosen without taking into account other factors than the ease of playing the set of notes, and sometimes the system is forced to choose alternative positions because of the errors in the MIDI files.

See appendix I for the comments made by the evaluators when evaluating the positions.

5.2.2 Prediction Accuracy, Precision and Recall

To calculate accuracy, precision and recall metrics, we used the confusion matrix in figure 5.3. We now explain, in the context of our problem, what are our true positives, false positives, false negatives and true negatives. The true positives are the positions chosen by the system correctly according to the ground truth positions. The system has identified the correct note and has chosen the same position that is in the ground truth for the equivalent note. The False Negatives correspond to the notes in the ground truth that the system did not identify, because they were not in the MIDI file, and they are also the notes that the system identified but chose a different position from the ground truth position. The False Positives are the extra notes present in the system, notes that the system has identified, because they exist in MIDI file, but do not exist in the ground truth, i.e., they are notes that do not exist in the music evaluated. They are

also the notes that the system identified but chose a different position from the ground truth position. True negatives do not exist in the context of our problem, since it is a binary problem, either the system chooses the correct position or it does not, true negatives do not make sense in this context.

The positions chosen by the system that differ from the ground truth positions belong to both false positives and false negatives. On the one hand, false negatives tell us that the system has, in this case, a position and the ground truth does not have it, because the chosen position differs from the right position. On the other hand, false positives tell us that the system does not have the respective position and the ground truth does, which is also what happens when a wrong position is chosen by the system.

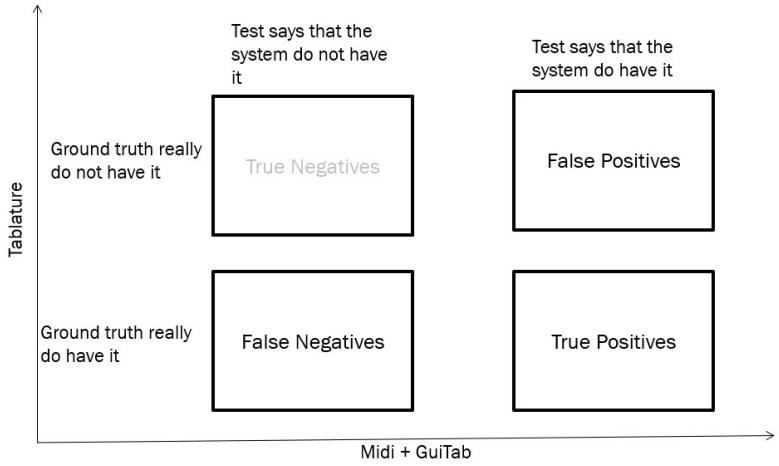

Figure 5.3: Confusion Matrix

The accuracy of the confusion matrix is given by the following formula:

$$accuracy = tp + tn/(tp + tn + fp + fn) \tag{5.1}$$

Figure 5.4 shows the accuracy of the 16 evaluated songs. The piece *Spanish Romance* has a much higher accuracy than the other pieces for two reasons. The first is due to the fact that the MIDI file was corrected manually. Extra notes were eliminated, missing notes added and note dependencies corrected. This correction allowed the system to come much closer to the choices that are in the ground truth. The second reason is due to the fact that there are a lot of loose notes in the music in question and our system, when choosing the best choices, benefits the use of loose notes.

The music *Dedicatoria* has an extremely low accuracy compared to the other pieces. This is due to dependency errors between notes, and to the choice of a different path of positions from the one those chosen by the ground truth. We know that the choice of positions is made taking into account the context in which they are inserted, so if a set of notes has positions different from the ground truth, the positions of the following notes will be different from the ground truth positions because, given the positions of the previous notes, they make more sense to be chosen by the system. An error in the choice of one position is usually not an isolated error, since it can propagate to other positions.

The accuracy of the remaining pieces is close to the average, which is 51%. These values are low but expected, considering the 20% of errors present in the MIDI files and considering that both the false positives and the false negatives contain the system's choice of wrong positions. Since each wrong position chosen by the system is counted twice, two other accuracy metrics were created and are going to be explained next. For now we present the precision and recall graphs where the values are much higher.

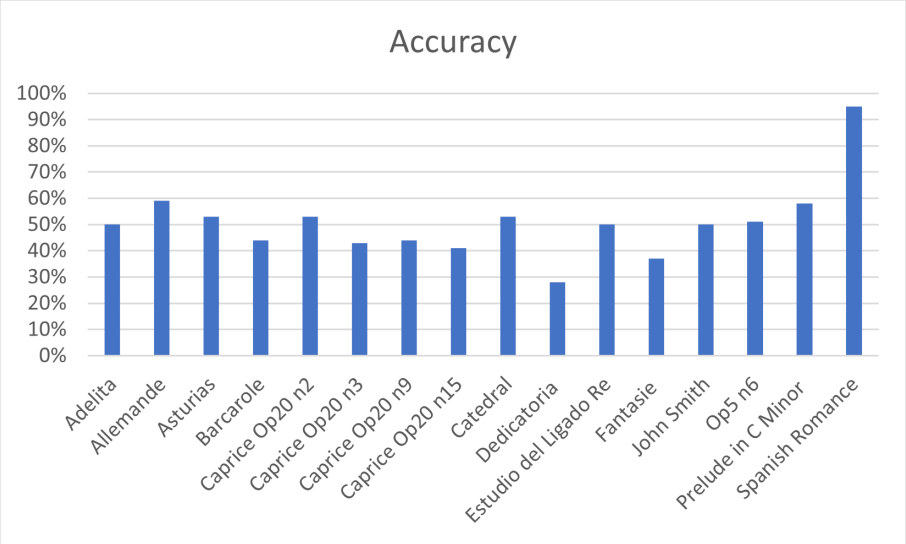

Figure 5.4: Accuracy Graph

The precision of the confusion matrix is given by the following formula:

$$precision = tp/(tp + fp) \tag{5.2}$$

Since the positions misplaced by the system are no longer counted twice in both precision and recall calculations, significantly higher values are expected. The difference between precision and recall, in the context of our problem, is that precision takes into account the extra notes in the MIDI file while recall takes into account the missing notes in the MIDI file.

The precision average is 67%, which means a big increase in comparison to the accuracy values as expected. Except for the two songs already mentioned above, the remaining pieces of music are close to the average, as can be observed in figure 5.5. This shows one of two things: either the % of wrong positions chosen by the system is similar for all the musical pieces, or when it is not, there is a high number of extra notes that makes the value close to the average.

The recall of the confusion matrix is given by the following formula:

$$recall = tp/(tp + fn) \tag{5.3}$$

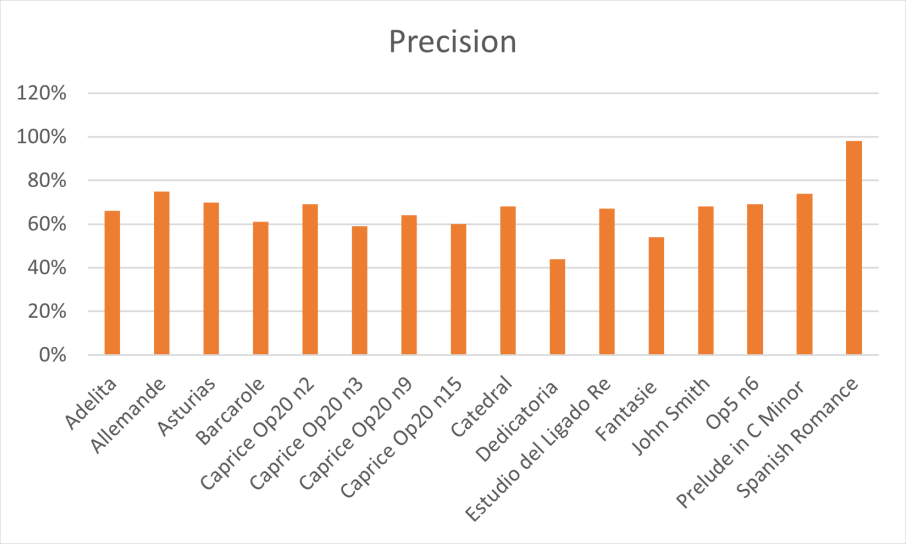

Figure 5.5: Precision Graph

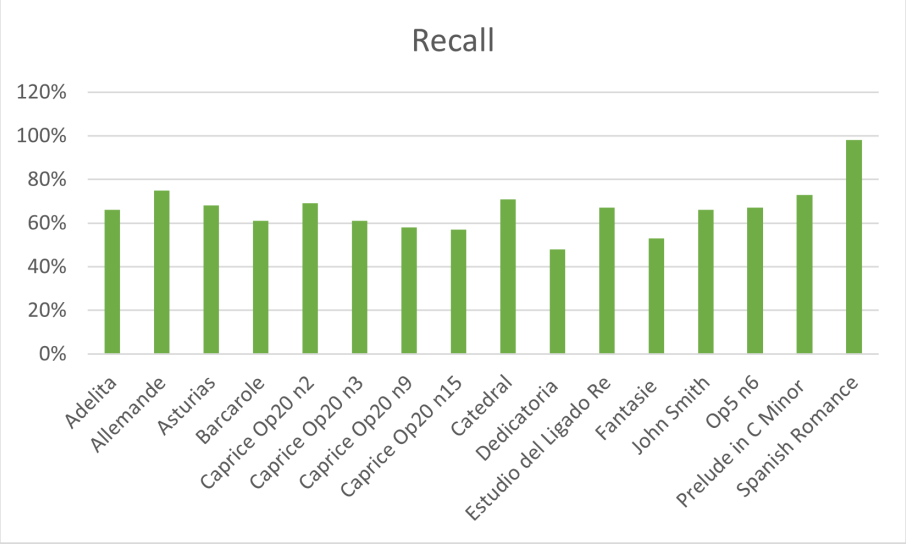

Figure 5.6: Recall Graph

The recall average is 66%, which is quite similar to the precision average, showing us that the number of extra notes is quite similar to the number of missing notes, and looking at both figures 5.5 and 5.6 we see that this is true for each song, since the values between precision and recall are very similar.

Since accuracy calculus is impaired by the repetition of the system's chosen wrong positions, which are either false positives or false negatives, it is not possible for us to conclude what is the weight of the system's chosen wrong positions and what is the weight of the first two types of errors in the MIDI files.

For this reason we define the prediction accuracy on a sequence of positions as the agreement between the ground truth positions and the ones chosen by the system, as formulated in the equation 5.4. $\sqcap$ is a binary function that returns 1 if the input is $\hat{0}$ and 0 otherwise. The summation takes into account all the positions that exist in the ground truth. After the sum is done, it is divided by all the notes present in the ground truth plus the extra notes identified by the system. We exclude extra notes from the summation because they are automatically wrong notes and would be undermining the number of correct positions identified by the system.

$$acc\hat{u}racy(\hat{p}, p) = \frac{1}{notes_{all}} \sum_{notes_{groundTruth}} \sqcap(\hat{p} - p) \tag{5.4}$$

Looking at the graphs presenting the metrics precision 5.5, recall 5.6 and $acc\hat{u}racy$ 5.7 we observe that the values are quite similar, thus concluding that the causes of the differences between the ground truth and the output of our system are the wrong positions chosen by the system and the wrong dependencies between notes. We also know that the existence of an extra note, which is present between correctly identified notes, can change the system's choice of positions, and the same is true for missing notes. An exhaustive manual evaluation would be required to understand the real weight of the existing errors in MIDI files. The solution would be to correct all MIDI files, but we are talking about thousands of notes and hours of work which was not possible to do during out thesis.

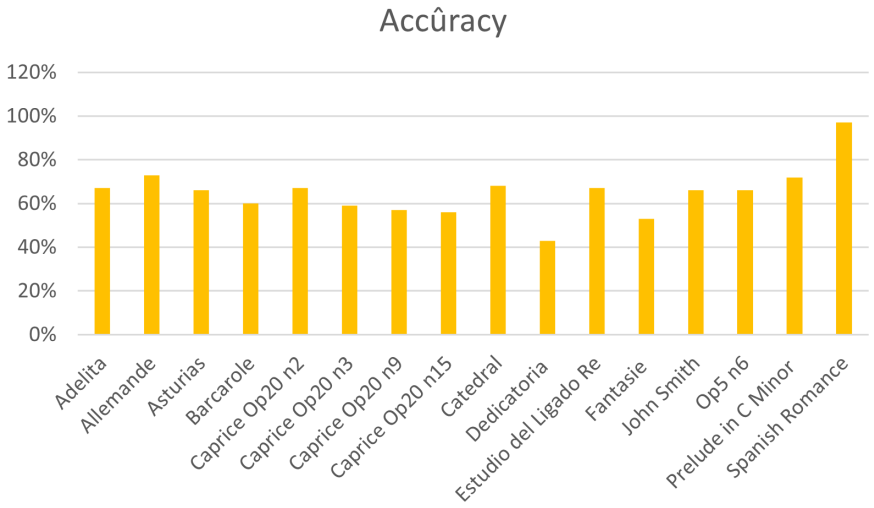

Figure 5.7: $Acc\hat{u}racy\ Graph$

Finally, another prediction accuracy was defined in the equation 5.5. This new accuracy does not take into account the first two types of errors present in MIDI files. That is, it does not account for the missing notes in the MIDI that are present in the ground truth and it does not account for the notes that are in the MIDI but are not in the ground truth. The note variable present in equation 5.5 is the intersection between the notes in ground truth and the notes present in MIDI files. Unfortunately, given the difficulty of identifying and correcting the third type of error in MIDI files, it is not possible for us to exclude it, but we have the notion that this error, since it is the most common one, undermines the results of the system evaluation.

$$\overline{accuracy}(\hat{p}, p) = \frac{1}{notes} \sum_{notes} \sqcap(\hat{p} - p) \tag{5.5}$$

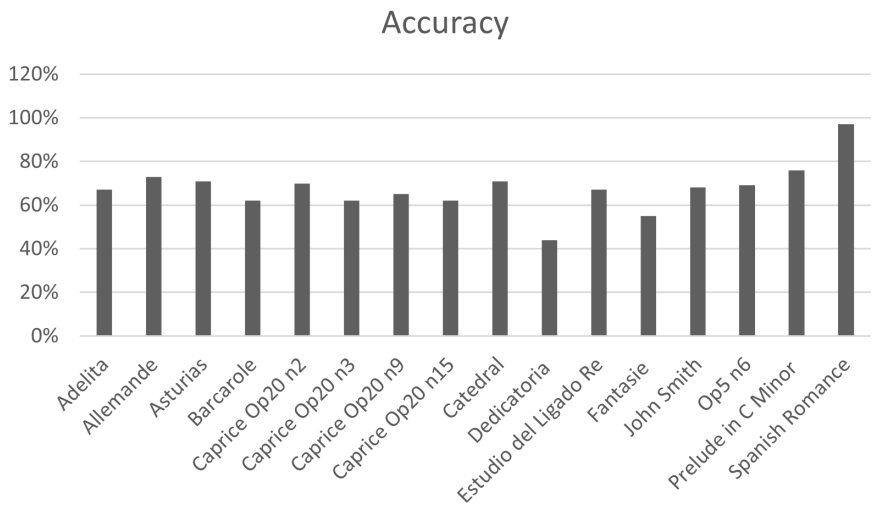

Figure 5.8: $\overline{Accuracy}$ *Graph*

Figure 5.8 shows the accuracy of the 16 songs without taking into account the first two types of errors present in the MIDI files. The average accuracy is 68% while the accuracy taking into account these two types of errors is 64%. This shows us that these two errors are not the errors that have more weight in the differences between the ground truth and the output of the system. Since these two errors are easy to identify, this is the most relevant accuracy and tells us how well the system behaves. The system behaves relatively well when choosing the canonical positions of the respective songs, however it can still present much more promising results. In the analysis made by the guitar players we noticed that, even though the chosen positions are different from the canonical ones, they are equally playable, which gives us confidence in the system's behavior. However, the canonical positions are chosen because in the context of the piece they are the most effective to play. Musical pieces as well-known as those evaluated are usually played in the same way by every professional guitarist. So we are aware that the system has sometimes failed to choose the most efficient positions and this is what we should focus on in the future.

5.3 Fingers Evaluation

To evaluate the fingers associated with the notes, three guitar players were asked to evaluate the associated fingers of the first 100 notes of a set of 5 pieces of music. We wanted the notes to be contiguous to each other because the choice of fingers depends on the context where the notes are inserted, i.e., given a note the following and previous notes are essential in the choice of the finger of the respective note. Given the repetition of patterns in the pieces, which is common in almost all pieces of music, the first 100 notes are enough to evaluate each piece of music. Each guitarist was also given the version before the fingering optimization, in order to see if the system can, in fact, by understanding the context in which the notes are inserted, optimize his or her fingers.

Two of our guitarists evaluators are different from the ones which evaluated the positions. Only the third evaluator who evaluated the note positions also evaluated the fingers. These two new evaluators learned guitar on their own. We wanted to find out if for guitarists with less formal education, the association also made sense. One of the guitar players has been playing electric guitar for many years and already has a lot of experience on stage. The second guitarist is still an apprentice, having just started to play guitar. The creation of tablatures is useful, as we mentioned at the beginning of our thesis, for people who have just started learning the guitar and we also wanted to understand how this guitarist adapted to the choices of the system. The third guitarist who also evaluated the choice of positions is the guitarist who has been teaching guitar for more than 20 years, as mentioned above.

We have always said that the choice of fingers is subjective, that it depends from guitarist to guitarist, and that the genre of music the guitarist usually plays also plays a role in the choice of fingers. One of the guitar players who evaluated the set of musical pieces does not play classical guitar but electric guitar, where he plays genres like blues and rock. He was asked to play the set of notes in figure 5.9. He played them using only the 1st finger, as it is common to do this when playing the blues in order to leave the other fingers free for blues techniques. However, on the classical guitar, it is unthinkable to use only the 1st finger to play this set of notes, showing that the musical genres, with their own techniques and ornaments also contributes to choose differently which fingers to use to press each note.

Figure 5.9: Set of notes from Spanish Romance

We did not try to make the guitarists' choice of fingers equal to the system's choice of fingers, since it is not a matter of choosing the right fingers or not, given the subjectivity of the choice. The goal of the evaluation was, first, to understand whether the optimization that the system performs in finger association given the context of the notes improves finger choice. Second, to understand whether or not the choice of associated fingers makes it impossible to play any notes. For example, if the same finger is associated with two notes directly dependent on each other, there is an impossibility to play that set of notes. Third, whether the system can tell when to use a barre or not, something that is quite common and important when playing classical guitar. Fourth, to understand how difficult it was to play the proposed set of pieces using the fingers chosen by the system.

Comparing the associated fingers before and after the optimization of the fingers it was unanimous that there was a huge improvement in the choice of fingers. After the optimization of the fingers taking into account the context where the notes are inserted, the difficulty of playing the respective pieces decrease a lot. This proves that the context where the notes are inserted is essential for the finger choice to be more appropriate. However, the guitarists concluded that there were certain parts that were not optimized, which the system may not have considered necessary to optimize. We recall that in the optimization section, moments that had more than four fretted notes were not considered for optimization. It should also be noted that there are cases where the the optimization of the fingers made it more difficult to pass from one set of notes to another than before the optimization. Although we did not have many cases like this, we had some.

Regarding finger choices that make it impossible to perform the note set, none were found by the guitarists. The system is always able to associate fingers so that the note set is always possible to play. The system is well able to tell when to make use of the barre and when not to. There are several examples of barres throughout the evaluated note set, and both the system and the guitarists chose to make use of the barre in the same situations, i.e. when they encountered the same note set. Given the high difficulty of execution of the pieces evaluated, it was not easy, on the part of the performers, to play them. But the difficulty did not come from the fingers used to press each note, but from the difficult execution of the pieces. In general, the system's choice of fingers is a correct one that makes it easier for the player to perform.

See appendix I for the comments made by the three evaluators when evaluating the fingers.

6 Conclusion and Future work

This first version of the *GuiTab* system has shown us, through system analysis, that it is possible to create a system, through the use of heuristics and rules, that can produce tablatures from a MIDI file. In section 1.1 we set out to answer some questions that we felt were pertinent. The first question was whether it would be possible to present results as good as those produced by humans. The analysis of the system showed us that yes, it is possible to create a system that presents tablatures as good as those produced by humans. Even if our system could not produce them, for all the reasons mentioned in the evaluation section, it showed us promising signs that it is possible to produce them.

The other question was about the subjectivity of choosing note positions in order to produce the best tablature, or whether there are several tablatures that are considered the best. The choice of the best tablature is actually subjective and has to do with the three factors mentioned in section 3.2. These are the biomechanical, cognitive and musical factors. So, for different guitarists these factors have different weights, thus showing that there is no best tablature, but rather a small set of tablatures that are considered the best. Our system, even though it did not always choose the canonical positions for the evaluated musical pieces, was able to produce tablatures that belong to this small set of tablatures.

6.1 Future Work

As future work it would be an improvement in the performance of the system to be able to understand when guitar techniques are used, as explained in section 2.2.1.2. These techniques require that the notes present in the respective technique are played on the same string, which automatically excludes a set of other possibilities. Another future work would be the complete correction of the MIDI files which would also greatly improve the performance of the system and allow it to be better analyzed.

We mentioned in the evaluation section that there are patterns in the use of the right hand that influence the choice of note positions. Given this, it would be interesting instead of only analyzing the behavior of the left hand to also analyze the behavior of the right hand and try to find out if there is a pattern in the respective music in order to try to maintain it when possible. It would also be interesting to make the system more dynamic for the user. To make it more dynamic one of the possible ideas would be to allow the user to change note positions manually. The user would choose the positions he/she wanted to change, and the system would check if they were possible to change and make the changes, i.e. it would check if changing the position changed both the positions of the surrounding notes and the fingers associated with the respective notes.

Finally, in section 2.1.1.3 we mentioned that classical guitar music is usually made up of sections. It would be an improvement for the system if it could find patterns in the music

so that it could associate the same positions with the notes of the repeated patterns without having to analyze them again.

Bibliography

Allen Mathews. (2017). *Spanish romance.* (https://www.classicalguitarshed.com/sm-spanish-romance//)

Allen Mathews. (2018). *Lágrima.* (https://www.classicalguitarshed.com/sm-tarrega-lagrima/)

Allen Mathews. (2019). *Asturias.* (https://www.classicalguitarshed.com/sm-albeniz-isaac/)

Allen Mathews. (2019). *Recuerdos da la alhambra.* (https://www.classicalguitarshed.com/sm-tarrega-recuerdos/)

Benetos, E., Dixon, S., Duan, Z., & Ewertr, S. (2019). Automatic Music Transcription: An Overview. *IEEE Signal Processing Magazine*, 20–30.

Bradford Werner. (2017). *Adelita.* (https://www.thisisclassicalguitar.com/wp-content/uploads/2015/05/Tarrega-Adelita-Free.pdf)

Chen, Y. H. (2020). Automatic Composition of Guitar Tabs By Transformers And Groove Modeling. *arXiv preprint arXiv:2008.01431*.

Cruse, H. (1990). On the cost functions for the control of the human arm movement. *Biological Cybernetics*, *62*, 519–528.

Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q. V., & Salakhutdinov, R. (2019). Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*.

Greg Burlet. (2018). *Frettable.* (https://www.frettable.com/)

Grout, D. J. (1960). . In C. V. Palisca (Ed.), *A History of Western Music.* New York City: W. W. Norton & Company.

Heijink, H. (1999). Take it easy: A model of left-hand fingering on the classical guitar. *Proceedings of the 1999 Conference of the Society for Music Perception and Cognition*, 96.

Heijink, H., & Meulenbroek, R. G. J. (2002). On the Complexity of Classical Guitar Playing: Functional Adaptations to Task Constraints. *Journal of Motor Behavior*, 339–350.

Masanobu Miura, N. H. M. Y., Isao Hirota. (2004). Constructing a System for Finger-Position Determination and Tablature Generation for Playing Melodies on Guitars. *Systems and Computers in Japan*, *35*(6), 755–763.

Mistler, E. (2017). *Generating Guitar Tablatures with Neural Networks* (Master of Science Dissertation). The University of Edinburgh, Edinburgh, Scotland.

Rosenbaum, D. A. (1991). Optimal movement selection. *Psychological Science*, 86–91.

Rosenbaum, D. A. (1995). Planning Reaches by Evaluating Stored Postures. *Psychological Review*, *102*(1), 28–67.

Rosenbaum, D. A. (1996). From cognition to biomechanics and back: The end-state comfort effect and the middle-is-faster effect. *Acta Psychologica*, *94*, 59–85.

Rosenbaum, D. A. (2001). Posture-based motion planning: Applications to grasping. *Psychological Review*, *108*, 709–734.

Scholes, P. (1938). . In *The Oxford Companion to Music.* Oxford University Press.

Tuohy, D. R. (2006). *Creating tablature and arranging music for guitar with genetic algorithms and artificial neural networks.* (Master of Science). The University of Georgia, Athens, Georgia.

Appendix A

Here we present the comments made by the three evaluators who evaluated the note positions. Comment from the first evaluator:

"In general, the system choices are quite playable and are occasionally even more comfortable than the canonical versions! However the canonical versions tend to make more sense in the overall context of the piece (for example, in Romance Anonymous the canonical version is markedly harder to play than the system version, but the system to achieve that ease breaks the very regular pattern of the right hand).

Honestly, I think what the system lacks to really be there is knowledge of the right hand, not the left hand! I think if the system knew that the right hand likes to arpeggiate adjacent strings in relatively regular patterns, that would help the system choose options that are more realistic and closer to canonical. But that is just my gut feeling, I might be wrong...".

Comment from the second evaluator:

"The alternative positions chosen by the system are always playable. Some are even more comfortable than the canonical ones. However, the most comfortable choices are not always the best choices, because it is necessary take into account other factors besides the ease of use of the left hand. As I tried to play the pieces in the alternative positions I felt that the canonical positions were more logical."

Comment from the third evaluator:

"Even if the alternative positions are playable, I do not think they add anything to the music. The canonical positions are more logical. However, I think that a lot of the alternative positions chosen by the system are due to the wrong dependencies between note. If the system chooses the best positions by taking into account the finger span and the movement of the left hand along the guitar neck, the canonical positions in relation to the alternative positions chosen by the system would make more sense."

Here we present the comments made by the three evaluators who evaluated the fingers association. Comment from the first evaluator:

"I compared the choice of fingers before and after the optimization made by the system and there is no doubt that there is an improvement after the optimization. Before optimization the use of finger 1 was often used when another of the 4 fingers would make more sense. Changing chords to others after optimization was much easier, since given that in positions that were repeated between the two chords the same finger was used and also because the system tries to use free fingers on the second chord. I did not detect any impossibility to use the associated fingers. One negative point is that sometimes the passage from one chord to the next could be optimized. Since I am not used to playing classical guitar I found it difficult to play the requested set of songs and sometimes I used different fingers than those chosen by the system."

Comment from the second evaluator:

"I had never thought about the question which fingers to associate with each note and how there can be several possibilities. I compared the choice of fingers before and after the optimization and no doubt the optimization makes the performance of the music easier. The set of songs is quite difficult to play, but this is not due to the choice of fingers, but to the difficulty of the pieces themselves. I have paid attention to the use of barres and the system can

identify them every time. I realized that the system can choose which fingers to associate with the notes better than I can."

Comment form the third evaluator:

"I know all the songs evaluated and most of the fingers are well associated. Much of the associated fingers before optimization do not make sense, yet after optimization, the new choices are correct. I have played and taught the songs in question, which made them easier to perform. I used barres in the same chords as the system. Some of my finger choices associated with the notes were different from the system. Sometimes because the system did not optimize optimally and sometimes any of the possibilities would be a good possibility. Overall the system manages to choose the best associated fingers."

II Appendix B

In the picture 1 is an example of MIDI file notation. The piano on the left tells us the note value, the size of the rectangles tells us the duration of each note, and the color of the rectangles tells us whether the notes belong to the bass, the melody or the accompaniment. To correct the MIDI file it is necessary to take a close look at this notation and compare it with the corresponding score.

Figure 1: Example of a MIDI File