Putting a Finger on Guitars and Algorithms

Computing Fingering Information for Guitar Tablatures

KONRAD ILZCUK
and PHILIP SKÖLD

KTH Computer Science
and Communication

Bachelor of Science Thesis
Stockholm, Sweden 2013

Putting a Finger on Guitars and Algorithms

Computing Fingering Information for Guitar Tablatures

K O N R A D I L C Z U K
a n d P H I L I P S K Ö L D

DM129X, Bachelor's Thesis in Computer Science (15 ECTS credits)
Degree Progr. in Computer Science and Media Technology 300 credits
Royal Institute of Technology year 2013
Supervisor at CSC was Roberto Bresin
Examiner was Sten Tärnström

URL: www.csc.kth.se/utbildning/kandidatexjobb/medieteknik/2013/ilczuk_konrad_OCH_skold_philip_K13018.pdf

Kungliga tekniska högskolan
Skolan för datavetenskap och kommunikation

KTH CSC
100 44 Stockholm

URL: www.kth.se/csc

Abstract

Guitar tablature is a notation system widely used by guitar players when learning to play songs. The notation consists of information about which string and which fret the guitarist should put his finger on. However, it does not normally contain information on which fingers should be put on which notes.

In this thesis we study existing methods related to incorporating fingering information alongside musical notations. Using this material, we define a set of factors that affect the guitar fingering and then design an algorithm for computing an optimal fingering for a tablature.

The final product is an algorithm that computes optimal fingering positions according to the relevant factors. The computed fingerings are evaluated by comparing them to how experienced guitarists play the given melody.

Our results indicate that it is possible to produce optimal fingerings algorithmically. This is a step forward in helping beginner and intermediate guitar players in their learning process.

Sammanfattning

Att Sätta ett Finger på Gitarrer och Algoritmer: Beräkning av Fingersättning för Gitarrtablaturer

Gitarrtablatur är ett notationssystem som används regelbundet av gitarrspelare för att studera och lära sig att spela låtar. Notationen innehåller information om var någonstans på gitarren man ska spela, det vill säga vid vilka strängar och vilka band man ska placera sina fingrar. Den innehåller däremot sällan information om vilket finger som ska användas för att spela respektive ton.

I den här uppsatsen undersöker vi existerande metoder relaterade till att berika musiknotationer med fingerinformation. Vi använder oss av den kunskapen och identifierar faktorer som påverkar fingersättningen och designar en algoritm som beräknar en optimal fingersättning. Sedan implementerar vi algoritmen och evaluerar resultatet genom att jämföra med hur professionella och erfarna gitarrister väljer att placera sina fingrar.

Resultaten visar tydligt hur beräknade fingersättningar ofta överennsstämmer med hur erfarna gitarisster spelar. Detta är ett steg framåt i att hjälpa gitarrister i deras tidiga inlärningsprocess.

Contents

Dedicated to the people working or studying at the Royal Institute of Technology.

Chapter 1

Introduction

For readers who are not familiar with terminology from computer science or music theory, we include a glossary as an appendix.

1.1 Background

There exist numerous ways of representing music. Among them we find those that are specifically made for the guitar to aid the guitar player in understanding and learning a song: notes, guitar tablature and chord diagrams, each way with particular advantages and disadvantages. Modeling, representation, generation, and evaluation of music notation are among the many problems that have interested researchers within computer science, and which have generated many applications.

1.2 Common Guitar Notations

1.2.1 Standard Music Notation

Standard music notation is often the prefered notation, as this represents music independently of instrument. This comes from the fact that it contains relative duration of a sound together with its pitch (Wikipedia, 2013).

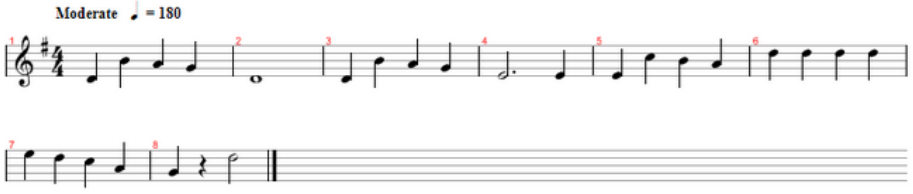


Figure 1.1. "Jingle Bells" represented in standard music notation.

1.2.2 Guitar Tablature

The Guitar Tablature is a notation popular among guitarists, mostly due to the fact that it takes relatively little time to learn to read it. It describes where on the guitar neck a finger should be pressed. The basic variants of tablature do not take any rhythmical values into consideration.

```
e|-------------------------------------------------------------------------------|
B|-------------------------------------------------------------------------------|
G|---9-7-----------9--7---------10-9-7---12--12--12--12-------------------------|
D|-5-----10----5--5------10-7-7-7-------------------------14--12--10--7------12---|
A|-------------------------------------------------------------------------10-----|
E|-------------------------------------------------------------------------------|
```

Figure 1.2. Sample text representation of a Guitar Tablature, "Jingle Bells", without rhythmical values.

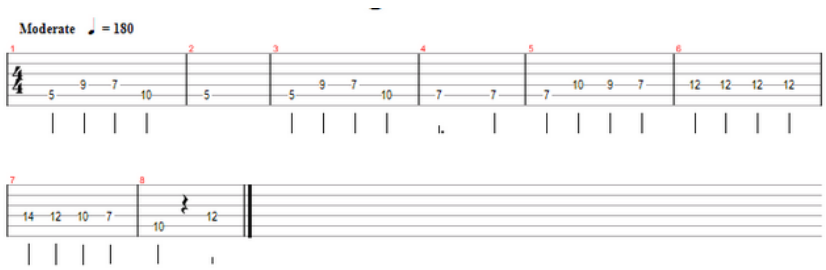


Figure 1.3. Sample graphical guitar tablature of the same song(Jingle Bells) enhanced with rhythmical values underneath the notes.

1.2.3 Chord Diagram

The chord diagram is a notation that represents chords - a combination of two or more notes. This notation often, but not always, includes which fingers should press which note.

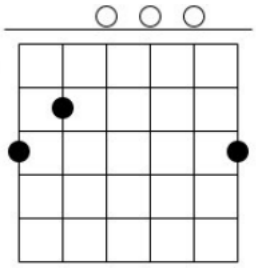


Figure 1.4. Chord Diagram representing an G Major chord (Guitar Instructions, 2013)

1.3 Guitar Fingering

Even when the guitar player has access to a notation for a song indicating where the notes are on the guitar neck, there are still many alternative ways to play that song. This is because each note can be played with any of the available fingers. The four playing fingers are usually referred to as finger 1, 2, 3 and 4, which corresponds to the index-, long-, ring- and little-finger respectively.

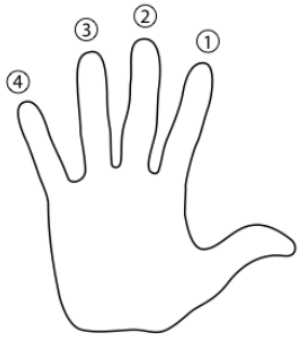


Figure 1.5. The playing fingers and their numberings (GuitarLessons.com, 2012).

The mapping of a note and the finger it should be played with is called a *guitar fingering*, or simply *fingering*. Fingerings are often presented with an integer from 1 to 4 placed next to, or under each note. This additional information is often referred to as LHF or RHF (Left/Right Hand Fingering).

```
e ------------------------------------------------------------------------------
B ------------------------------------------------------------------------------
G ----------9---7-------------------------9-----7----------------------10--9---7---
D ------5-------------10--------5-----5---------------10--7---7---7-----------------
A ------------------------------------------------------------------------------
E ------------------------------------------------------------------------------

      1   3   1     4         1     1   3     1     4   1   1   1     4   3   1
```

Figure 1.6. A representation of a tablature, "Jingle Bells", together with the fingering below each note.

Chapter 2

Problem Statement

The number of possible fingerings for a melody is vast - 4^N, where N is the number of notes in a song (Rutherford, 2009). To stress how large this number is, a song with just 15 notes would have more than a billion fingering possibilities. This introduces a problem - many beginner or intermediate guitar players have difficulty knowing which fingerings are best. Anecdotal evidence and our own experience as guitarists show that questions like "how do I play this?", "why do my fingers hurt?" or "is this impossible?" often arise when people play a melody in a way that is not the best.

Fingering information can be of great help for musicians. An alternative to fingering notation would be private tutoring or recorded tutorials where a professional guitarist explains the exact fingering to the tutee. Additionally, it is possible to manually add fingering notation to both new and existing tablatures if one knows how, but it is a very repetitive and slow process. Because of this, a large number of available tablatures lack fingering information, making it difficult for some less-experienced players to learn their favorite songs. Moreover, at the time of writing (April 2013), there is no popular software (Informer Technologies, Inc., 2013) available to guitar players that helps them compute fingering information.

We will refer to the problem of algorithmically computing the best fingerings of a guitar tablature as the *Optimal Fingering Problem (OFP)*.

Chapter 3

Aims

3.1 Purpose

The main goal is to determine if it is possible to algorithmically provide fingering information for existing tablatures.

Our aim is to prove or disprove whether an implementation with disjunct rules[1], that reflect the complexity of guitar playing, is a sufficient method to solve the OFP.

The purpose of the thesis is to improve the learning process for beginner and intermediate guitar players by providing state-of-the-art approaches to solve this problem algorithmically.

3.2 Hypothesis

The difficulty of guitar playing depends on a limited set of identifiable factors. We define and list these factors so that they later can be used as the basis for a solution to the OFP. Our main research questions are:

- Is it possible to compute good fingerings by implementing guitar *complexity factors*[2] as disjunct rules?
- Is there a limit on what can be achieved algorithmically and will the solutions be adequate to significantly improve the song learning experience for guitar players?
- Will the algorithm be suitable for complex as well as simple melodies?

[1] Stand alone rules, that work independently of one another.
[2] Factors that make up the difficulty of guitar playing.

3.3 Scope of Work

We have limited our work to handle melodies where only one note is played at a time and without considering any time and rhythmical aspects. Also, we have only implemented those complexity factors that were necessary to conclude whether the method works or not. Factors that affect guitar playing, but have nothing to do with how the guitar player organizes his/her fingers on the guitar neck have been intentionally omitted. Within this thesis the algorithm has been our main focus and so we have excluded graphical and user-friendly aspects from this study.

Chapter 4

Literature Research

One problem that has been considerably analyzed is the problem of generating a tablature from standard musical notation (Tuohy and Potter, 2005; Radicioni and Lombardo, 2007). Much of the research that can be found on the Tablature Generation Problem (TGP) is relevant in the OFP. Furthermore, the complexity of guitar playing, and specifically the term *complexity factor* (Heijink and Meulenbroek, 2002), has proven very fruitful in solving the OFP. Analyzing advantages and disadvantages to existing approaches to the LHF and similar problems (Tuohy and Potter, 2005; Radicioni and Lombardo, 2007) aided us in our own solution.

Categorizing music notation in terms of fingering can be done using chords, melodies and mixed passages (Radicioni and Lombardo, 2007). We cover only one of these parts - the melodies - in our study. Furthermore, the authors suggest that each of the three distinct categories requires a different approach when evaluating which fingers are to be prefered.

There are also advanced techniques such as string bending[1], slides[2] or hammer-on and pull-offs[3] that can be used. Rutherford (2009) suggests that such techniques also have an impact on the fingering.

[1] a very common technique performed (most often) on the electric guitar in which the player usually uses his third finger to bend the string upwards while in the process creates a smooth transition from one note to another.

[2] A smooth transition generated by sliding a finger from one note to another.

[3] A technique of playing notes in a way that makes them sound linked to each other.

4.1 Complexity Factors

To create a solution to the OFP, we need to evaluate how hard or easy a fingering is to play. Heijink and Meulenbroek (2002) hypothesise three factors (biomechanical, cognitive and musical) that would reflect the complexity of finger movements in guitar playing, which they refer to as *complexity factors*. The three factors they identified were the need to reposition the hand during a tone sequence, the position on the guitar neck and the required finger span to play a sequence comfortably.

4.2 Existing Approaches

There exists a number of possible ways of computing a fingering. What all of these approaches have in common is that, at some point, the solution must evaluate which alternative is the best. The practical aspects of guitar playing must be reflected in some way in the solution. Each solution is thus highly dependent on how the algorithm treats the practical aspects. The same holds true for the TGP.

There has been extensive research within the area of tablature generation. Many different approaches have been suggested, some more sophisticated than others. Most of the concepts within them are applicable for the OFP. An easy to implement approach is to model the problem as a graph and then finding a solution using a path-finding algorithm[4] between two nodes in a graph. This is exactly what Radisavljevic and Driessen (2004) propose. Dynamic programming[5] has also been used to solve the fingering problem (Radisavljevic and Driessen, 2004).

Tuohy and Potter (2005) argue that the nature of the TGP makes *exhaustive searches*[6] tedious and impractical. The same concern is present in the OFP as the number of fingerings grows exponentially with the number of notes in a song. For the TGP, Touhy and Potter (2005) suggests using genetic programming, an approach that could be used in the OFP as well. Genetic programming is a kind of *heuristic*[7] that mimics evolution and survival of the fittest. In both concepts the complexity factors would be reflected in what is called a *fitness function*[8].

We have also contacted the company Arobas Music (Appendix B.1), developer of the popular tablature software called "Guitar Pro", and discussed implementing such functionality into their product. They are already working on this themselves (February 2013), which proves that the topic is very current at the time of writing. However, it is unclear what approach they will use. There have been remarks on

[4] An algorithm that scans a graph to find the best path between two points.

[5] A method for solving complex problems by breaking them down into simpler, overlapping, subproblems[12].

[6] A method of computing a result in which all possible combinations are examined before the final result is delivered.

[7] A technique for finding an approximate solution when classic methods fail to find any exact solution[14]

[8] A function used to indicate how good a solution is, relative to given aims.

how badly the tablature generation within that particular software performs (Tuohy and Potter, 2005).

Additionally, as mentioned in section 2 (Problem Statement) there is no software that assigns fingers algorithmically to melodies (Informer Technologies Inc., 2013). However, we found that "Guitar Guru" is a product that comes closest to this, because it presents a fingering for songs, but these have been manually assigned instead of computed and are available to buy from their database of around 2500 songs. Lastly, there are also programs that display fingerings for guitar chords, but again - these have been manually assigned.

Chapter 5

Method

We model the problem as a graph and use this as the basis upon which to construct the algorithm. The graph contains information about notes, fingers and transitions between pairs of {Note,Finger}. This is very intuitive to understand and an easy way of implementing the underlying data structure.

We identified a set of complexity factors that affect fingering and implemented them as disjunct rules that are used to *weight*[1] the finger-transitions. The process of weighting involves each rule scanning the graph, and then scoring each *edge*[2], according to how desirable each transition is. Finally we use a path-finding algorithm to determine the optimal fingerings for the entire melody, by traversing the graph from the first note to the last.

5.1 Complexity Factors

As we introduced in the section 4.1, Heijink and Meulenbroek (2002) use the term complexity factors to refer to factors that determine the complexity of left-hand finger movements in guitar playing. The same idea can be used when discussing the complexity of fingering transitions, i.e each possible fingering between two notes. We will use the term complexity factors as factors that represent *the complexity of a fingering transition.*

There are two stages of work to address the complexity of guitar playing. First, we identify the complexity factors. For instance, the required finger-span of a fingering is such a factor, as it clearly affects the difficulty of that fingering. By difficulty or complexity of a fingering, we mean a characteristic of that fingering rather than the difficulty that a guitar player may experience. Secondly, after we identify the set of complexity factors, we design concrete rules that describe a way to implement those factors, building upon the work of Heijink and Meulenbroek (2002).

[1] (verb) to assign a score, to prioritize.

[2] A link between two nodes in a graph. In our case, the edge represents a transition from one note to another.

5.2 Concept

Since our implementation contains multiple phases, we use a range of different data structures and algorithmical steps to read, process and output our data.

The tablature is represented as a matrix where each row is an individual string, each column represents a point in time and the cells contain information about the fret positions. This can be read from left to right while maintaining the notes in chronological order. To reiterate, our study does not consider time aspects. The tablature source does not affect the way we represent it. MusicXML file, Guitar Pro file or ASCII file, etc. could all be used. For simplicity we have decided to read input from ASCII[3] files.

```
E|----------
B|----------8
G|----------
D|--5--6---
A|----------
E|----------
```

Figure 5.1. An example ASCII-tablature that will be translated to a matrix.

null	null	null
null	null	8
null	null	null
5	6	null
null	null	null
null	null	null

Figure 5.2. A matrix representation of tablature.

Because we will be using a Optimal Path Approach, we now model the problem as a graph problem where we begin by translating the data from the matrix to a graph. Each note is represented by 4 vertices, representing the possible fingerings of that particular note. Then each transition between two note-fingerings is represented by an edge, resulting in 16 edges for each pair of notes.

[3]Files containing plain, readable text.

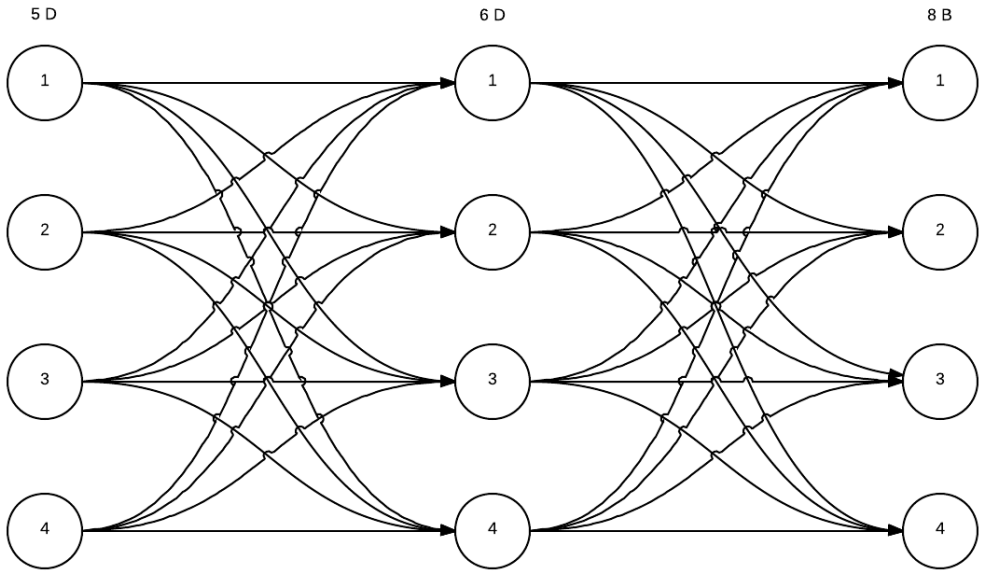


Figure 5.3. A graph with vertices for all possible note-fingerings. The columns shows the fingering options(1,2,3,4) for the notes "5 D" (fifth fret on the D-string), "6 D" and "8 B" respectively. The edges represent every possible transition between two notes.

The next step is to weight each edge. This is done using the *Visitor Design Pattern*[4] in which each rule visits each edge, weighting the edge with a *scoring function*, thus indicating how beneficial that transition is. The beneficiality of a transition indicates how easy it is to go from playing the given note with a specific finger, and from there move on to another note with perhaps the same, or a different finger. This often means that the musician switches between fingers that are convenient to use if that results in a smooth transition between fingering positions and as little as possible hand movement along the guitar neck.

When the weights are assigned, we find an optimal path through the graph, selecting the overall best transitions, thus generating (according to our rules) an optimal fingering for the tablature.

The final output is the original tablature, beneath which we find our computed ten best fingerings. A proposed finger is written directly under the corresponding note for good readability.

[4]Visitor Design Pattern - A way of separating an algorithm and the object it operates on.

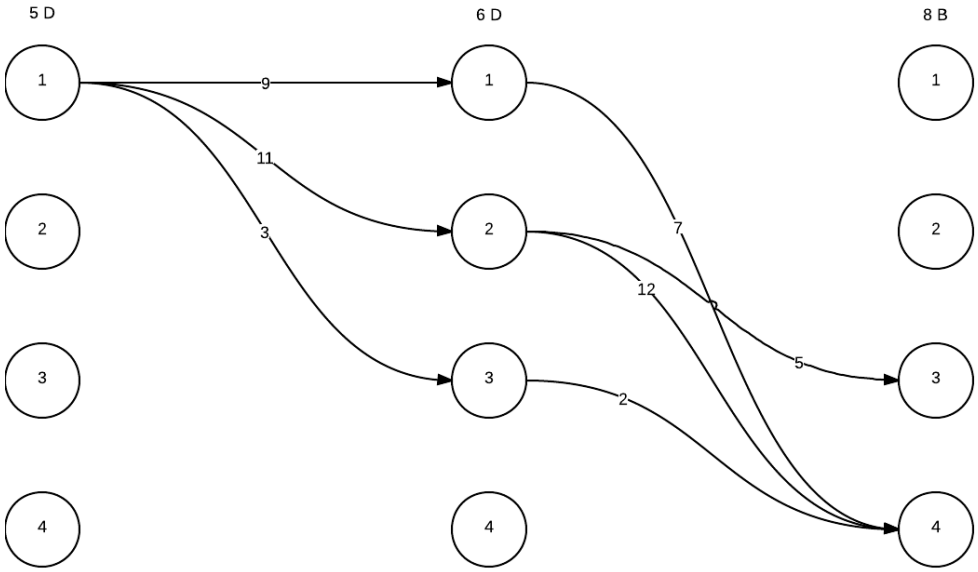


Figure 5.4. A graph showing how some edges have been assigned with a weight. In this case a high weight indicates a "good" transition.This graph has been simplified for readability.

5.3 Algorithm Complexity

The *computational complexity*[5] arises from the time taken to find the optimal path in the graph, as all other steps in the algorithm are done in linear time in relation to the amount of notes. A simple way to find the optimal path in the weighted graph is to perform an exhaustive search. This approach would give the algorithm a complexity of $4^{(N-k)}$, where N is the total number of notes and k is the number of open notes, thus making it *exponential*[6]. However, as the graph is a *directed acyclic graph*[7] there exists a proof that an optimal path can be found in linear time (Rivest et al., 2001).

5.4 Limitations

This approach of modeling the problem as a graph to find an optimal path, only works for melodies. However, working with complexity factors, and implementing them with disjunct rules is fully applicable to other approaches and works for more than just melodies.

5.5 Tools

We have chosen a high level object oriented programming language - C# - as our tool to implement the algorithm. The main reason for this is that the C# language allows for rapid code development and easy code maintenance.

5.6 Evaluating Results

The output of the algorithm is compared with how experienced guitar players, and composers of a melody, play the melody themselves. This is done to analyze any differences and draw conclusions as to why they occur.

[5] An indication of how well an algorithm performs with respect to time. More or less: how fast it computes in relation to the amount of data provided in the input stage.

[6] Exponential Complexity indicates that the growth rate is proportional to the function's current value[13]

[7] A graph in which there exists no path(sequence of edges) from any vertex such that it creates a loop back to the same vertex.

Chapter 6

Result and Discussion

In this section we present the results of our study, which include the complexity factors we have identified, which of those we have chosen to implement and how our algorithm performed for a number of different melodies. Each design concept and algorithm result is followed by a brief discussion.

6.1 Algorithm Performance

The number of fingerings is exponential, and our implementation runs in exponential time and space. With this running time, the algorithm manages to compute fingerings for melodies with up to 11 non-open notes[1]. With an optimization, where paths with no potential are ignored early, it can handle melodies with up to 19 notes. The upper bounds arise because the computer runs out of memory (as illustrated in table 6.1).

However, using a path finding algorithm with better computational complexity would increase this threshold significantly and at the same time produce the same results.

[1] these results were achieved on an Asus ux31a[7]

Number of notes	time [ms]	time (optimized) [ms]
1	2.00	1.75
2	1.75	2.00
3	2.00	2.00
4	2.50	2.25
5	2.50	2.25
6	9.00	2.50
7	28.8	4.00
8	133	6.50
9	610	12.3
10	2746	26.0
11	12209	47.0
12	out of memory	106
13	out of memory	251
14	out of memory	521
15	out of memory	779
16	out of memory	1616
17	out of memory	3553
18	out of memory	7498
19	out of memory	11476
20	out of memory	out of memory

Table 6.1. The table demonstrates an average running time (four measurements per each number of notes, rounded down) for the non-optimized version and the optimized version of our algorithm. Both of them increase exponentially after test cases with more than five notes. The time for the non-optimized version grows by a factor of approximately 4. This means that a melody with 20 notes would (if possible) take almost 4 hours to compute. The optimized version shows a much better performance. The factor by which the time to compute increases is only around 2, but is slowly approaching 4 as the number of notes increase. For 20 notes it would (if possible) compute the results in around 23 seconds.

6.2 Complexity Factors

We identified a set of factors which reflect the complexity of guitar fingering transitions i.e complexity factors, as explained in section 4.1. We have then partially addressed these factors with concrete rules that weight different fingering transitions in order to produce a suggestion of an optimal fingering, according to the optimal path approach.

The identified complexity factors are more specific than those used by Heijink and Meulenbroek (2002). Below, we present the identified complexity factors, one by one, and discuss why we have chosen them.

6.2.1 Required Finger Span

The *note distance* is the distance between two notes, expressed in how many frets there are between them. Depending on how great this distance is some fingerings will be harder or easier to play.

For instance a transition from the 4th fret to the 5th requires a finger span of 1 fret. This would be easiest to play with two adjacent fingers ({1, 2}, {2, 3},{3, 4} using the finger enumeration explained in section 1.3). Furthermore, the same adjacent fingerings become harder to play as the notes move further apart from each other (thus increasing the note distance).

6.2.2 Finger Properties

Different fingers have different strengths which is determined by their physical properties and how often they are used. This is most notable in the little finger, which is generally weaker than other fingers. The stronger the finger, the easier transition involving that finger will be.

6.2.3 String Change

Fingering works differently when we change the string we are playing. A same-finger transition can work very well between two frets on the same string, but not necessary between the same frets on different strings.

6.2.4 String Distance

String distance indicates how much the player must move his/her hand in order to reach out for the new string. A fingering may work well for a transition between two adjacent strings but not between two strings that are further apart.

6.2.5 Available Fingers

When playing a new note it is preferable to use those fingers that are not currently assigned to a fret, or that have not been recently assigned.

6.2.6 Articulation

Many complexity factors reflect *biological* and *mechanical properties*, like the finger span or finger strength, but there are also musical factors that affect the complexity of guitar playing (Heijink and Meulenbroek, 2002). *Note articulation* occurs when individual notes, or transitions between notes, are played with a *technique* in order to produce a very specific, desired musical effect (Wikipedia, 2013). Many such articulations affect the fingering (Rutherford, 2009; Heijink and Meulenbroek, 2002).

6.2.7 Neck Position

A transition may be easy at one position on the neck, but as soon as the player tries to map the same fingers to notes that are further away, it becomes more difficult. This comes from the fact that distances between frets gradually become smaller towards the top of the neck. Heijink and Meulenbroek (2002) also mention this as a complexity factor.

6.2.8 Time, rhythm

Rhythmical values play an important role in determining a fingering. The slower the melody, the more freedom the player has in the choice of fingers, as he/she has enough time to use any of them. However, in very fast sequences the choice of fingers narrows, as we cannot afford certain transitions due to the delay they cause.

6.2.9 Ascending and Descending Sequences

Knowing the direction of a sequence of notes is important for the player when planning fingerings for the next steps in the melody. In both ascending and descending progressions the player is highly unlikely to re-use fingers that he or she has recently used. The player first uses the available fingers and then, if necessary, repositions the hand in the direction of the sequence.

6.2.10 Repeaned Sequences

Sometimes it is easier for the player to play a fingering he/she has played before on various parts of the neck, even if they are not the best ones from the perspective of the other complexity factors. Recurring patterns are sometimes better to play the way they were first played, instead of needing to memorize individual fingerings for each occurrence.

6.3 Implemented Rules

After identifying the set of complexity factors, we implemented only those that were necessary to conclude whether our hypothesis was correct or not. The subset of these factors was expressed in the algorithm as disjunct rules. Each rule provides a score that modifies the weight of the given edge, where the edge represents the transition between two notes with a suggested fingering.

Below, we describe the scoring system, followed by how our rules work and why they were implemented.

6.3.1 Scoring System

The impact a rule has on a transition is subjective. There is no way of knowing what exact score the rule ought to give. Instead, the scoring system works as follows: an enum, Score, has the discrete terms: full, good, average, lower-average, little, very little and none. Each term has an integer value between 0 and 100, a percentual expression relative to the defined maximum score each rule can give a transition. This way, each rule will express how good a transition is using those terms. The enum that represents the scoring alternatives is presented below.

Enum Score: {
None = 0,
VeryLittle = 1,
Little = 5,
AvgLower = 25,
Avg = 50,
Good = 75,
Full = 100
}

To exemplify how this works - an Average Score (Score.Avg) would score 200 if the maximum score was 400.

A constraint to the defined maximum is that it needs to be high enough for the results to be distinguishable. For example, a maximum of 3 gives an average score and a good score the same value. This means that we lose accuracy because transitions of different qualities get the same weight. A maximum of 100 proved empirically to be sufficient, allowing for a good degree of freedom.

6.3.2 Distance Rule

The first, and also most influential rule, is the *distance rule*. It addresses almost entirely the complexity factors: *required finger span* and *ascending and descending sequences*. The importance of this rule comes from the fact that the required finger span is the best indicator of whether a fingering is convenient to play.

The rule begins by calculating the natural finger span and the required finger span. The values indicate the distance between two notes and the distance between the two fingers, respectively. Both distances are expressed in frets. Depending on the outcome of the note distance, we classify the behaviour of this rule into two cases:

Short Range

This is when the player can reach the note distance without needing to reposition his/her hand. The difference between the required finder span and the natural finger span determines the score. The larger the difference, the more difficult the transition is, because it requires stretching the fingers unnaturally. This difference translates into a weight using our scoring system - the smallest difference scores a maximum, while the biggest scores zero points.

Long Range

When the next note is out of the player's reach, the rule values all fingers equally since the whole hand must be repositioned.

function DISTANCERULE(edge)
  $noteDistance \leftarrow edge.to.fret - edge.from.fret$
  $fingerDistance \leftarrow edge.to.finger - edge.from.finger$
  if *ShortRange* **then**
    if *FingersCrossing* **then**
      return
    else
      $result \leftarrow abs(noteDistance - fingerDistance)$
      $score \leftarrow Translate(result)$
      $edge.DoWeight(score)$
    end if
  else if *LongRange* **then**
    return
  end if
end function

The rule also partially covers the ascending and descending sequences complexity factor. This is because negative note distances correspond to descending patterns and positive ones to ascending.

6.3.3 String Change Rule

The idea of examining the difference of the natural finger span and required finger span(see section 6.3.2, distance rule) works very well, as long as concerned with the same string. However, as soon as the transition involves a string change, the quality of results decline, especially string changes to the same fret generate almost unplayable fingerings. This is the reason why we introduced the String Change Rule.

The String Change Rule is divided into two major cases: transitions to the same fret and transitions between different frets:

Transition to Same Fret

In this case it is generally best to choose higher or lower fingers depending on if the string change is to a higher or lower string. If the transition is to a higher string, transitions to higher fingers: 1 to 2 or 2 to 4 etc. are preferred. If the transition is to a lower string, transitions from higher to lower fingers are preferred instead.

Barring is also an option if it is a string change to the same fret. The String Change Rule always favours barring with the first finger especially if the strings are adjacent.

Transition to Other Fret

When the transition is to another fret or there is no string change involved at all the String Change Rule does nothing. In both of these cases the Distance Rule is sufficient and produces very good results.

```
function StringChangeRule(edge)
    if not StringChange then return
    end if
    if TransitionToSameFret then

        //easy to bar with the first finger on adjacent strings
        if ToAdjacentString and BarWithFirst then
            edge.DoWeight(Score.Avg)
        end if

        //best to bar with first finger
        if BarWithFirst then
            edge.DoWeight(Score.Little)
        end if

        if TransitionToLowerString then
            if TransitionToLowerFinger then
                edge.DoWeight(Score.Good)
            else
                edge.DoWeight(Score.Little)
            end if
        else if TransitionToHigherString then
            if TransitionToHigherFinger then
                edge.DoWeight(Score.Good)
            else
                edge.DoWeight(Score.Little)
            end if
        end if
    elsereturn
    end if
end function
```

6.3.4 Little Finger Rule

The Little Finger Rule covers a part of the complexity factor regarding the fingers' biomechanical properties. Implementing this rule allows the utilization of the natural attributes of each finger, in this case, their strength.

This simple, yet useful, rule favors transitions that go to any other finger than the little finger, by giving a *little* weight bonus to all fingers except the little finger. Often, it is definitely preferable to use the little finger, but in those cases other rules will compensate for it.

function LittleFingerRule(edge)
  if *not* $TransitionToLittleFinger$ **then**
    $edge.DoWeight(Score.Little)$
  end if
end function

6.3.5 Slide Rule

This rule partially addresses the complexity factor Articulation. It was chosen as a proof that articulation could be parametrized and included in the algorithm.

The rule works by favoring transitions to the same finger, whenever slides are possible.

function SlideRule(edge)
  if $TransitionToSameString$ **then**
    if $TransitionToSameFinger$ **then**
      $edge.DoWeight(Score.Little)$
    end if
  end if
end function

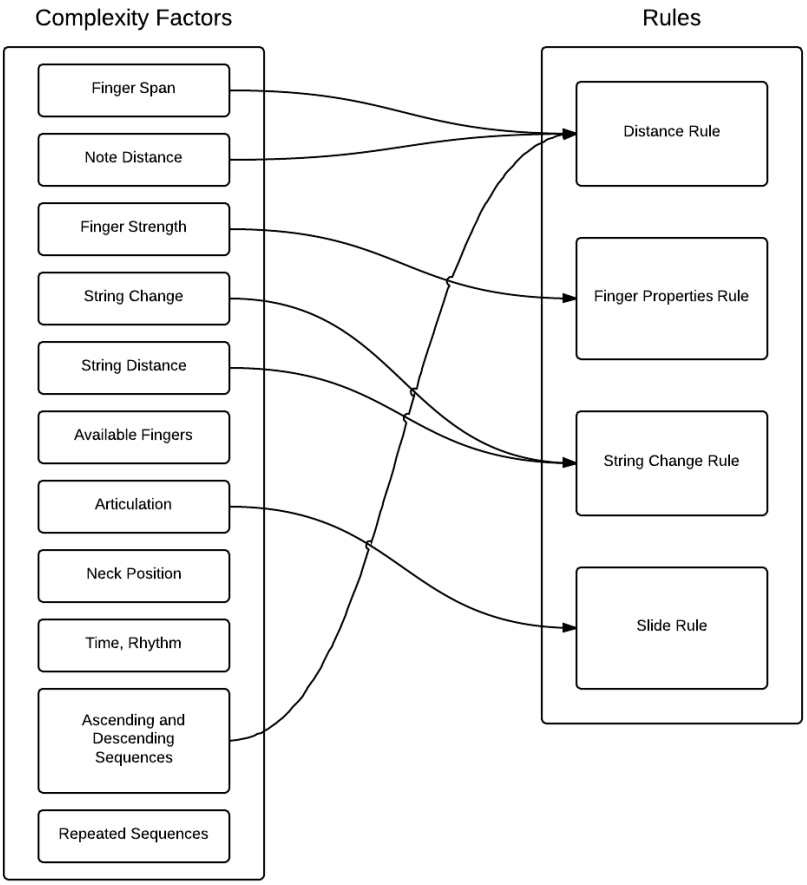


Figure 6.1. The identified complexity factors with the mapping to the implemented rules.

6.4 Algorithm Result Presentation

This subsection demonstrates the result our algorithm produced for a range of different input melodies together with a discussion part after each one of them. The melodies we used have been chosen carefully to test different aspects of the algorithm. The list below shows the songs from which we have extracted the melody inputs.

1. Racer X - Technical Difficulties
2. Deep Purple - Sometimes I Feel Like Screaming
3. Led Zeppelin - Black Dog
4. Led Zeppelin - Ocean
5. AC/DC - Thunderstruck
6. Steve Vai - Jibboom
7. Steve Vai - For The Love Of God
8. Guthrie Govan - Uncle Skunk
9. C-Major Sweeping Pattern, taken from an instructional video
10. A. Vivaldi - L'estate Presto
11. A Minor Pentatonic Scale

The results for all 11 melodies are presented in the following form:

- The reason why the song was chosen,
- The tablature for the examined melody,
- The output of our algorithm - fingerings,
- The real fingering - the one played by the composer or other experienced guitarist,
- Brief comment regarding the similarities or differences between our fingerings and the real fingering,
- Highlighting differences for better readability,
- Discussion regarding the reasons for differences or similarities.

F stands for *"Fingering"*. The fingerings will be listed one after another in descending order, meaning that the first from top is considered "best" by our algorithm, and the subsequent ones are sometimes equally good, or just a little worse.

RF stands for *"Real Fingering"* which reflects how the composer or other experienced guitarist plays the melody.

6.4.1 Racer X - Technical Difficulties[2]

Chosen because:

It does not utilize any advance techniques, such as: bends, slides and tappings. It contains some notes that come after each other on the same fret but on adjacent string.

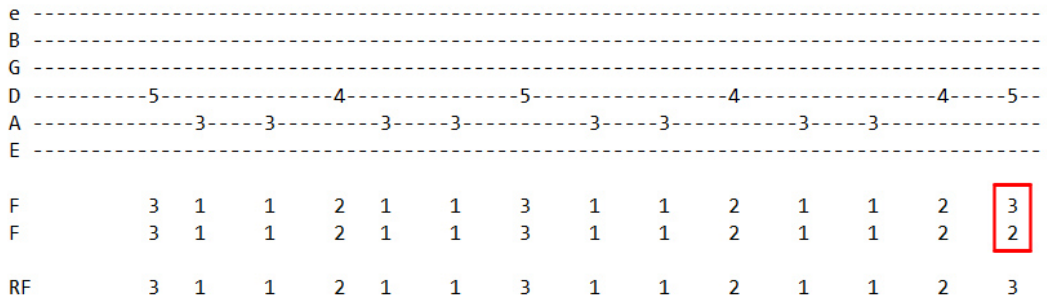


Figure 6.2.

Comment:

The real fingering (RF) is exactly the same as our optimal solution (best fingering). The second solution, which uses a slide on the last note, is also considered optimal by us. The reason for this is that the differing finger in the transition to the last note is comfortable to play with a slide. The second best result can be accredited to the Slide Rule (section 6.3.5).

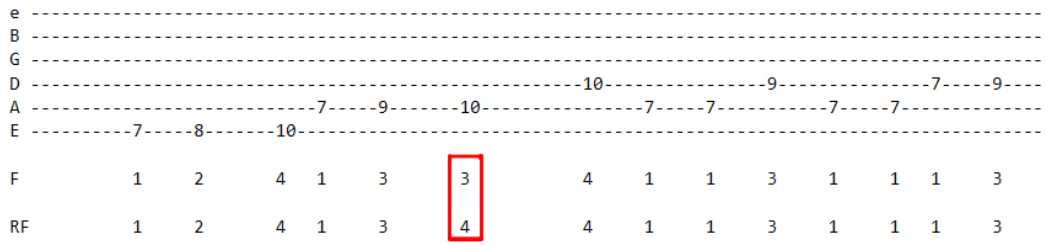


Figure 6.3.

Comment:

The real fingering is the same as our optimal solution with one difference on the transition between the adjacent 10ths.

[2]source: see reference [23]

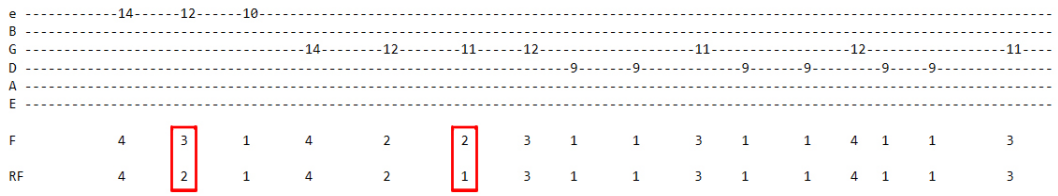


Figure 6.4.

Comment:

Same as our optimal with two minor differences. The first twelfth fret is played with the second finger instead of the third, as our algorithm suggests. The first eleventh fret is played by Paul Gilbert[3] with the first finger, instead of the second.

Discussion:

Overall, the computed fingerings are good. Our optimal fingering in the first sample (Figure 6.2) is exactly what Gilbert uses when he plays it.

The second melody (Figure 6.3) differs because in our implementation a transition on the same fret, but between adjacent strings with the same finger, is considered technically more difficult than to use another finger. Although this may be true for many average players, Gilbert achieves his speed partially due to how he plays this part.

Also, we estimate that transitions to the same fret on adjacent strings are easier to do with the first finger than with others, because we assume that this would mean that the finger was barred. This is why our result differs from when Gilbert makes such a transition with his little-finger but not when he bars with other fingers, mainly because he does not bar his little-finger but instead makes a very quick jump.

Gilbert plays the third melody (Figure 6.4) with the 12th fret with the second finger, whereas our optimal solution says the third would be best. The difference arises as the algorithm does not consider neck position. On higher frets, playing this part with the 2nd finger is easier than with the third, however from a note and finger distance perspective(which we take into consideration), the third finger performs better.

[3]Racer X's guitar player, who can be seen in the source video for "Technical Difficulties".

6.4.2 Deep Purple - Sometimes I Feel Like Screaming

Chosen because:

Contains some string-skipping, places where the player could, but doesn't have to slide from and to notes leaving open the chance to interpret how to play it. We suggest the RF for this melody.

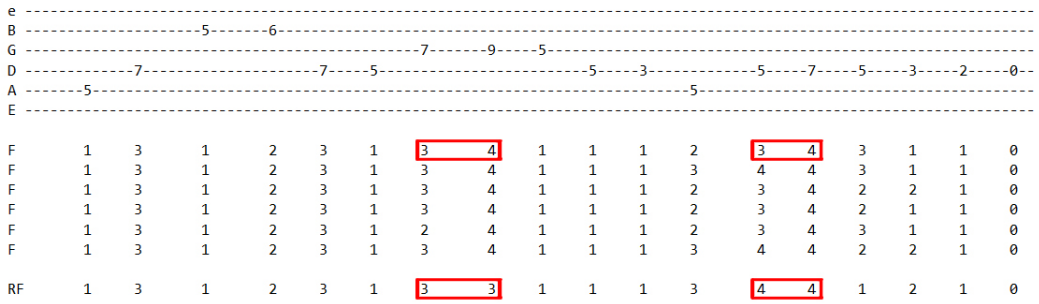


Figure 6.5.

Comment:

Many of the computed fingerings match the preferred fingering of guitar players (RF). There are differences on just a few notes (highlighted parts above), where the RF suggests sliding.

Discussion:

The algorithm makes little consideration of slides, which gives rise to the differences with the RF as highlighted. The difference here is due to our estimation of the importance, or influence, of the slide rule. Some songs are more "slidy". A more detailed input with notation for sliding would eliminate these problems. Increasing the influence, however, of the slide rule gives the preferred (optimal) result.

6.4.3 Led Zeppelin - Black Dog[4]

Chosen because:

Firstly it contains string changes both to the same fret and to other frets, as well as a string-skip from the A to the D-string. It also contains a repeated pattern that is played many times on different parts of the neck.

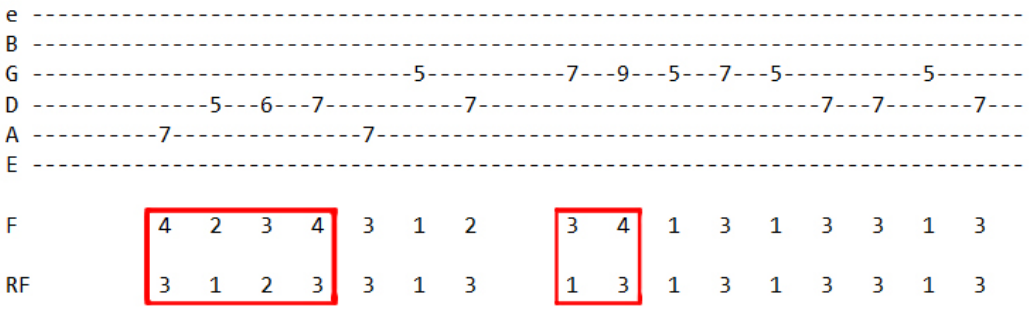


Figure 6.6.

Comment:

Our result differs from how it is played. The first bit is usually played with fingers 1, 2 and 3, with the third finger barred on the transition between the two 7ths. Also, when the melody goes up to the 9th fret it is usually done with the third finger, followed by a jump, but our top solutions suggests using the fourth.

Discussion:

The first part is erroneous because the algorithm favoured playing without barring. This results in a fingering that is more difficult to play, mostly because of the finger strengths of fingers 2, 3 and 4. This suggests that there might be more to finger strength than we originally anticipated.

The reason our algorithm suggests the second part should be played with the fourth finger, and not with the third, and also including a jump (as in the original), is that our algorithm always assumes that minimum hand-movement is optimal. But in this case, there is a specific sequence that is played repeatedly on a different part of the neck, which is why these sequences in the original are played with the same fingerings even though it requires more hand movement. It is also worth mentioning that the jump is possible because the melody is not played using legato[5].

[4]Source: see reference [24]

[5]A technique of playing notes in a way that makes them sound linked to each other.

6.4.4 Led Zeppelin - Ocean[6]

Chosen because:

It is a technically easy song where the RF is very intuitive.

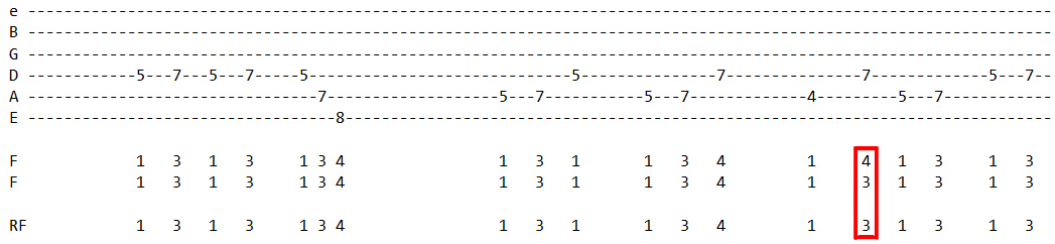


Figure 6.7.

Comment:

The results have been obtained by assuming edges scoring less than 50% in total, from all rules, are not important enough to be retained. Our second result is the same as the RF.

Discussion:

Our second solution is the exact same as the RF. However, the two top solutions that we suggest have been weighted the same, meaning that the algorithm cannot tell whether one would be better than the other. In reality, how you would play this comes down a preference, but neck position does also have an effect here. The reason that we weight them the same is because we assume that it is as hard to stretch fingers as to play them tightly together than their natural span.

What is notable in these results is that in order to cope with melodies this long we have applied an approximation where we assume that in a good fingering we will never use transitions that are not very good, thus getting rid of many "weak" edges in the graph. The fact that the results are still very good indicates that the song is technically not very difficult and that the weak finger-transitions are obviously not very likely to be used at all.

[6]Source: see reference [20]

6.4.5 AC/DC - Thunderstruck[7]

Chosen because:

Every other note is an open note, which is a special case which we have not taken into consideration.

```
e ------------------------------------------------------------------------------
B --------------0---4---0---7---0---4---0---7---0---5---0---8---0---5---0---8---
G ------------------------------------------------------------------------------
D ------------------------------------------------------------------------------
A ------------------------------------------------------------------------------
E ------------------------------------------------------------------------------

F               0   1   0   1   0   1   0   1   0   1   0   1   0   1   0   1
F               0   2   0   2   0   2   0   2   0   3   0   2   0   3   0   3
F               0   2   0   2   0   2   0   2   0   3   0   2   0   1   0   2
F               0   2   0   2   0   2   0   2   0   3   0   2   0   1   0   3
F               0   2   0   2   0   2   0   2   0   3   0   2   0   2   0   1
F               0   2   0   2   0   2   0   2   0   3   0   2   0   2   0   2

RF              0   1   0   4   0   1   0   4   0   1   0   4   0   1   0   4
```

Figure 6.8.

Comment:

None of our top solutions are close to the RF.

Discussion:

It is apparent that the results are not optimal. The reason is that we only look at single transitions, and each note can be played with four fingers that are equally good in case we go to or from an open note[8]. In order to correct this, we would have to look ahead instead of just looking at one transition at a time.

[7]Source: see reference [18]

[8]Open string/note - A note that is played without the involvement of any fret-pressing.

6.4.6 Steve Vai - Jibboom[9]

Chosen because:

Its an example of a song with open strings that we can solve (contrary to the previous example: section 6.4.5, Thunderstruck)

```
e ------------------------------------------------------------------------
B ----------5---3---0---5---3---0-------------------------------------------
G ----------------------------------4---2---0-------------------------------
D ------------------------------------------------4---2---0-----------------
A ----------------------------------------------------------4---2---0------
E ------------------------------------------------------------------------0--

F          3   1   0   3   1   0   3   1   0   3   1   0   3   1   0   0

RF         3   1   0   3   1   0   3   1   0   3   1   0   3   1   0   0
```

Figure 6.9.

Comment:

The algorithm produces the exact same result, as how Steve Vai plays the song.

Discussion:

The fingers 3 and 1 are chosen instead of 2 and 4 because they're stronger. It is also an example where the same pattern is played repeatedly with the same fingering, and the algorithm recognizes that as the most convenient way to play it.

What is notable in this example is how each open-string-note effectively makes the problem space four times smaller. As we recall from section 5.1 the computational complexity is $4^{(N-k)}$, where N is the total number of notes and k is the number of open note. Each time k is incremented by one, the problem space gets divided by four.

[9]Source: see reference [17]

6.4.7 Steve Vai - For The Love Of God[10]

Chosen because:

The melody is very articulate, meaning the player is strongly encouraged to "decorate" the notes with different techniques. We do not take many of these techniques into consideration. It also has some string changes and also requires the player to reach for the higher frets.

```
e ---------------------------------------------------------------------------------------------
B --------------------10----12------12----------17------15------12------------------15----10-
G --------------7-------------------------------------------------------12------16-------------
D ----------9---------------------------------------------------------------------------------
A ---------------------------------------------------------------------------------------------
E ---------------------------------------------------------------------------------------------

F          3   1    3    4      4         4      3      1    1      4    3    1
F          3   1    3    4      4         4      4      1    1      4    3    1
F          3   1    4    4      4         4      3      1    1      4    3    1
F          3   1    4    4      4         4      4      1    1      4    3    1
F          3   1    4    1      1         4      4      1    1      4    3    1

RF         3   1    3    3      1         3      3      1    1      2    1    1
```

Figure 6.10.

Comment:

These results did not match how Steve Vai plays the melody. He chooses to stretch to the 16th fret with his third finger, while our solutions play that with the fourth. Another noticeable difference is that the two adjacent 12-fret-notes are played by him exactly as one note, but with two different fingers.

Discussion:

The differences from our best computed results come from the fact that the melody is played in a unique and possibly unintuitive way which uses an interesting combination of fingers. Vai uses his method to express a specific feeling with the melody and to adjust for the next notes. One example are slides, where our algorithm suggests slides only when it is convenient but in this particular composition the slides appear as artistic expression. This means that, in the melody the slides are there because the artist likes the sound of them, not due to how comfortable it is to perform them. Such ways of playing are sometimes referred to as “advanced techniques” and may alter fingerings Rutherford (2009).

[10]Source: see reference [16]

6.4.8 Guthrie Govan - Uncle Skunk[11]

Chosen because:

it incorporates a very fast, advanced arpeggio[12] and slides.

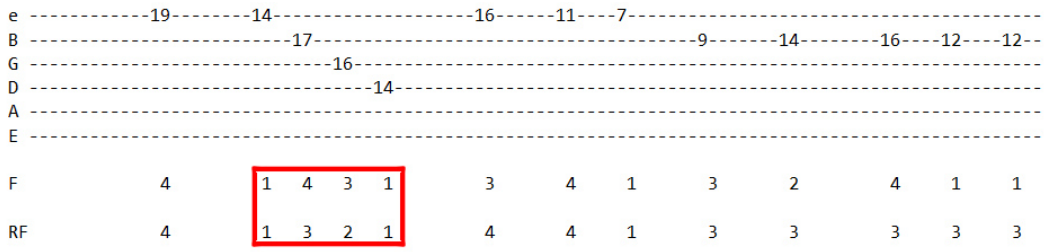


Figure 6.11.

Comment:

Our first result differs slightly from how the melody is usually played. Firstly, our it mainly suggest using the fingers 4, 3 and 1 in the arpeggio whilst it is preferably played with 3, 2 and 1. Secondly we do not favour sliding as much, thus giving a different finger suggestion for the second part.

Discussion:

The reason the arpeggio differs is because the time aspect is not considered in the algorithm. The transitions from the first note through the arpeggio is too fast, so using the little-finger in the arpeggio is not an option because there is not enough time to move it there. Taking this into account would require looking at the time aspect, as well as looking at more than two notes at a time in order to determine what is feasible and what is not.

The second part is interesting because we produce a better result by increasing the influence of the sliding rule as a whole. It does not suggest the exact same fingering as Govan but comes closer. In the transitions 16 to 11 and 9 to 14 it is especially important to slide, which our modified algorithm also suggests.

[11] Source: see reference [21]

[12] A musical technique where notes in a chord are played or sung in sequence, one after the other, rather than ringing out simultaneously[10].

6.4.9 C-Major Sweeping Pattern[13]

Chosen because:

t is an arpeggio over a chord. This pattern is often used for fast solos and is also the basis for improvisation.

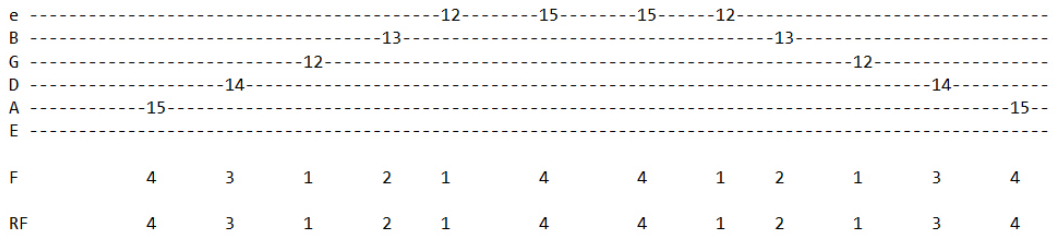


Figure 6.12.

Comment:

Our optimal solution is the one that the guitar tutor plays.

Discussion:

This is a case where there is little space for articulating notes. The RF therefore corresponds very well with the rules by which the algorithm finds the best set of transitions.

[13]Source: see reference [22]

6.4.10 A. Vivaldi - L'estate Presto

Chosen because:

it is a melody that is played fast, has many string changes and therefore requires the player to use the right fingers in order to achieve the speed as in the original. We suggest the RF for this song.

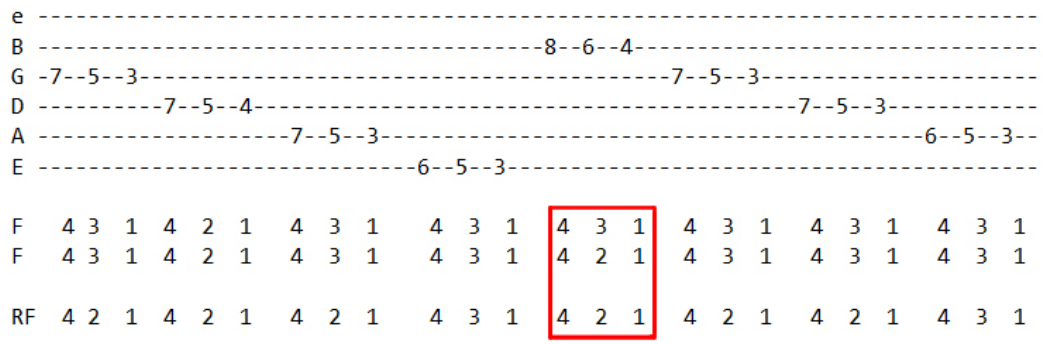


Figure 6.13.

Comment:

The top results are variations of each other in places where they can vary. In those places the choice of a specific finger is just as good as the choice of any other and therefore all fingers are weighted the same way.

Discussion:

The optimal fingering is debatable in this case, both because using either the second or third fingers to play the playing the middle notes in the descending triples is equally comfortable. Most people however use the second finger. Our top results are variations of that, showing the use of both second and third fingers. If we implemented a rule that takes the current neck position into consideration, the optimal answer would suggest the 2nd finger.

6.4.11 A Minor Pentatonic Scale[14]

Chosen because:

Scales are the key to understanding how different tones and harmonies work together. They are often used as an exercise to improve technique and speed. Chords are built up on the basis of scales.

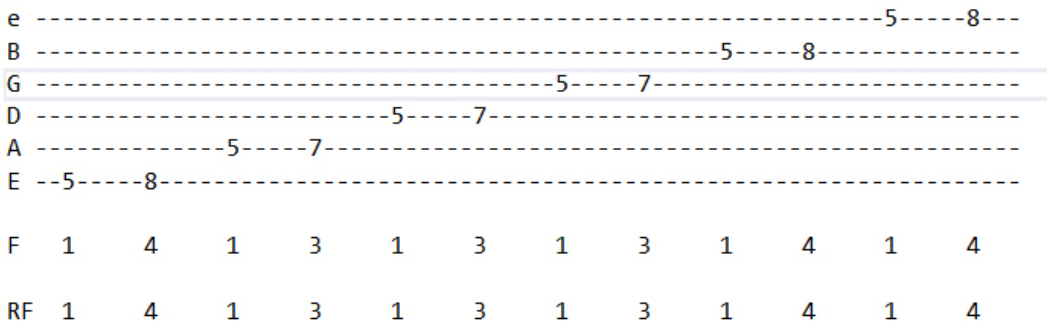


Figure 6.14.

Comment:

Our optimal fingering is exactly the same as the real fingering.

Discussion:

We have primarily implemented complexity factors that are biomechanical, and achieved good results for many melodies. This is especially apparent in examples like this one, where the melody is simple in both cognitive and musical terms (Heijink and Meulenbroek, 2002); the biomechanical complexity factors reflect the fingering constraints accurately.

[14]Source: see reference [19]

Chapter 7

Conclusion

To summarize our discussion, we have identified the major characteristics that influenced the results.

7.1 Analysis of Results

7.1.1 Repeated Pattern and Cognitive Factors

Our results show that it is not always preferable to play the most convenient fingering, as we assumed it would be. As also identified by Heijink and Meulenbroek (2002) there are times when cognitive factors play a role in choosing guitar fingerings. Some melodies, for instance Black Dog (section 6.4.3) contain repeated sequences where it is cognitively easier to use the same fingering every time the same sequence is played, because of the natural process of memorizing a fingering. The guitarist remembers one fingering and applies this to several instances of the same pattern, even though it is biomechanically more complex for some of those instances.

7.1.2 Decoration and Articulation

We conclude that there are major differences in how songs should be played if articulations and note *decorations*[1] are considered. Some melodies are very artistic and expressive, where individual notes and transitions have a specific character and feel, while other melodies are more technical. This means that there are two distinct types of melodies to consider when calculating a fingering:

Linear, simple melodies:

- The most comfortable transitions are regarded as exceptionally good with respect to many rules.
- The least comfortable transitions are regarded as exceptionally bad with respect to many rules.

Complex/artistically expressive melodies:

- Most transitions are considered effective with respect to some complexity factors, but uncomfortable with respect to others.
- Many transitions are classified merely as acceptable and the contrast between and their weights differ only marginally.

Linear melodies result in considerable differences between the alternatives for fingering and make it very easy and also effective to completely ignore many fingering alternatives. Intuitively, there is often one fingering that is the obvious choice. Very technical playing, for instance the C-Major Sweeping Pattern or Racer X's "Technical Difficulties", fall under this category.

Complex melodies require that most edges are retained, because only few scored so low that they could be omitted without affecting the fingering result in an overly negative way. Paths are weighted more or less equally, as there are no contrast between the different alternative fingerings; most transitions are both good and bad with respect to different rules. For instance we might have five fingerings that are, technically, as effective.

In order to deal with more complex melodies the simplest solution would be to handle more expressive input notation. For instance there are notations for slides, tappings and bendings. Extended notation will eliminate many fingerings that are simply undesirable.

[1] Note decoration - a general term by which we mean everything that can be done with a single note, more than simply playing it. This includes: switching between fingers while still holding the same note, vibrating it in a subtle and special way etc.

7.1.3 Biomechanical Characteristics

There are individual biomechanical constraints that influence fingerings. Finger strength or hand size, for instance, have an impact on which fingering each guitarist finds most convenient. Some guitar players may not be able to stretch their fingers as far as other players. Another example would be a player with an injured finger, who would probably not wish use it extensively. A possible solution to this is to take into account physical attributes of the hand and the guitar of the user.

7.1.4 Individual preferences and technical skills

Another factor that is apparent from the results is that individual skill is also a factor. We assumed that it was not easy to jump to the same fret on an adjacent string for instance, yet Paul Gilbert (Racer X) did not find it a problem at all, hence the difference in fingerings. This suggests that personal preferences are important. Techniques, like barring, sliding, bending or stretching are in reality performed slightly differently by different guitar players. For instance, skilled guitar players are in general better at bending tones with the fourth finger than intermediate or beginner guitarists are. The average guitar player finds it harder to bend with the fourth finger than with any other finger.

7.2 Possible Improvements

7.2.1 Applicability

Given that the number of fingerings grow exponentially, our simple path-finding implementation is not practical. Users would, and should, demand to be able to get computed fingerings for longer melodies. To make the algorithm applicable for practical use the optimal path should be calculated with better time complexity. We suggest a dynamic programming approach which would run linear time. The existence of such was mentioned in chapter Method: Algorithm Complexity (4.1) . Furthermore, we want to emphasize the importance of developing a user friendly interface for a commercial version.

7.2.2 Tablature Explicitness

As our results suggest, a more explicit tablature notation would in many cases dramatically improve the results. The fact that many available tablatures do contain more information than that we have considered is a strong reason to, in the future, also implement these functionalities. This is especially relevant for notation of bends and slides.

7.2.3 Complexity Factor Implementation

It is apparent that fingerings differing from the RF arise when some of our complexity factors are not considered. In our solution, only six out of the ten complexity factors we identified are implemented. Implementing all of them would significantly improve the results.

Chapter 8

Summary

To conclude, our hypothesis appears to be accurate. Implementing complexity factors as disjunct rules gives qualitative results for the Optimal Fingering Problem.

The independent operation of the rules allows for a holistic evaluation of fingerings and is particularly suited for simpler melodies. However, implementing more Complexity Factors and allowing users to specify articulation preferences would make it possible to also handle complex melodies really well.

Solving the Optimal Fingering Problem in this fashion is very much suitable for a commercial software for generating fingerings to melodies. Deploying such software would greatly aid guitarists in their learning process by allowing them to receive fingering information for any melody of their choice and in many cases save up on private tutoring.

Appendix A

Glossary

ASCII File - File containing plain, readable text.

Arpeggio - A musical technique where notes in a chord are played or sung in sequence, one after the other, rather than ringing out simultaneously[10].

Barre - a technique of playing notes on the guitar where one finger is put on many strings of the same fret enabling the guitarist in the process to play many tones without the necessity of moving the hand.

String Bending - a very common technique performed (most often) on the electric guitar in which the player usually uses his third finger to bend the string upwards while in the process creates a smooth transition from one note to another.

Computational complexity - An indication of how well an algorithm performs with respect to time. More or less: how fast it computes in relation to the amount of data provided in the input stage.

Directed Acyclic Graph - A graph in which there exists no path(sequence of edges) from any vertex such that it creates a loop back to the same vertex.

Disjunct Rules - Stand alone rules, that work independently of one another.

Dynamic Programming - A method for solving complex problems by breaking them down into simpler subproblems. Applicable to problems exhibiting the properties of overlapping subproblems[12].

Edge - A link between two nodes in a graph. In our case, the edge represents a transition from one note to another.

Exhaustive search - A method of computing a result in which all possible combinations are examined before the final result is delivered.

Exponential Complexity - Exponential compelexity is when the growth rate is proportional to the function's current value[13].

Fingering Transition - A transition between two notes described by which finger the first note is played, and with which one the other one is played.

Fitness Function - A function used to indicate how good a solution is, relative to the given aims.

Graph - A data structure where objects are connected by links(edges).

Guitar Neck - The part of the guitar where the player puts his fingers on top

of strings to determine the pitch.

Heuristic - A technique for finding an approximate solution when classic methods fail to find any exact solution[14].

High Level Object Oriented Programming Language - A programming language with a high level of abstraction, hiding many computer architecture details.

Legato (Hammer-on / Pull-off) - A technique of playing notes in a way that makes them sound linked to each other.

Note decoration / Note Articulation - a general term by which we mean everything that can be done with a single note, more than simply playing it. This includes: switching between fingers while still holding the same note, vibrating it in a subtle and special way etc.

Open String Note/Open Note - A note that is played without the involvement of any fret-pressing.

Path-finding Algorithm - An algorithm that scans a space(or in our case: a graph) to find the best path between two points.

Sliding (glissando slide) - A smooth transition generated by sliding a finger from one note to another.

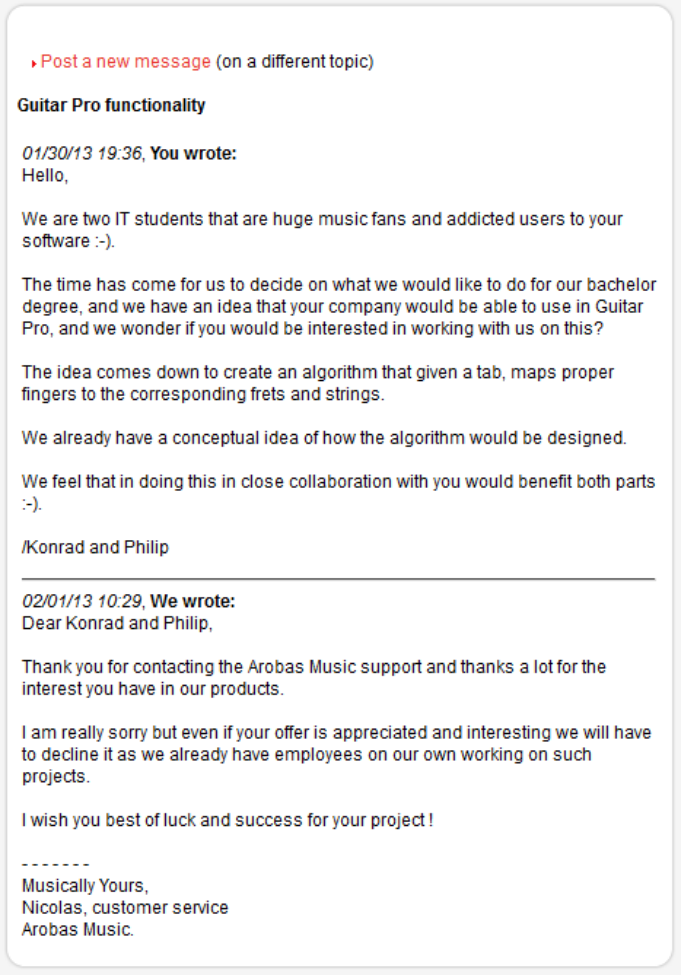
Sweeping - a technique in which the player "sweeps"(plays) through many strings by going up and down the guitar neck. This often implies the player has good coordination between his left and right hand. A sweeping pattern is often used in fast solos, chord arpeggios and improvisation.

Visitor Design Pattern - A method of separating the algorithm from the objects it operates on.

Weighting - (verb) to assigning score, to prioritize.

Appendix B

Arobas Music Email



Post a new message (on a different topic)

Guitar Pro functionality

01/30/13 19:36, **You wrote:**
Hello,

We are two IT students that are huge music fans and addicted users to your software :-).

The time has come for us to decide on what we would like to do for our bachelor degree, and we have an idea that your company would be able to use in Guitar Pro, and we wonder if you would be interested in working with us on this?

The idea comes down to create an algorithm that given a tab, maps proper fingers to the corresponding frets and strings.

We already have a conceptual idea of how the algorithm would be designed.

We feel that in doing this in close collaboration with you would benefit both parts :-).

/Konrad and Philip

02/01/13 10:29, **We wrote:**
Dear Konrad and Philip,

Thank you for contacting the Arobas Music support and thanks a lot for the interest you have in our products.

I am really sorry but even if your offer is appreciated and interesting we will have to decline it as we already have employees on our own working on such projects.

I wish you best of luck and success for your project !

Musically Yours,
Nicolas, customer service
Arobas Music.

Figure B.1. Email conversation with arobas music.

Bibliography

[1] T.H. Cormen, C. E. Leiserson, R.L. Rivest, C. Stein, 2001. Introduction to Algorithms, Second Edition. The MIT Press. Available at: http://ftp.tudelft.nl/TUDelft/oilie/Birmingham/Introduction%20to%20Algorithms.pdf

[2] Heijink, H., & Meulenbroek, R. G. J. (2002). On the Complexity of Classical Guitar Playing: Functional Adaptations to Task Constraints. Journal of Motor Behavior, 34(4), pp 339–351. Available at: http://dx.doi.org/10.1080/00222890209601952

[3] Radicioni, D., and Lombardo, V. (2007). A Constraint-based Approach for Annotating Music Scores with Gestural Information. Constraints, 12(4), 405–428. Available at: http://dx.doi.org/10.1007/s10601-007-9015-y

[4] Radisavljevic, A., and Driessen, P. (2004). Path difference learning for guitar fingering problem. Proceedings of the International Computer Music Conference (Vol. 28). sn. Available at: http://www.ece.uvic.ca/~peterd/papers/PDL_paperICMC2004_ver9.PDF

[5] N. T. Rutherford (2009). FINGAR, a Genetic Algorithm Approach to Producing Playable Guitar Tablature with Fingering Instructions. Available at: http://www.dcs.shef.ac.uk/intranet/teaching/public/projects/archive/ug2009/pdf/aca04ntr.pdf

[6] Tuohy, D., and Potter, W. D. (2005). A genetic algorithm for the automatic generation of playable guitar tablature. Proceedings of the International Computer Music Conference (pp. 499–502). sn. Available at: http://www.cs.uga.edu/~potter/CompIntell/Tuohy_Potter_Tablature_GA.pdf

[7] ASUSTeK Computer Inc., 2013. ASUS_ZENBOOK_UX31A. Available at: https://www.asus.com/Notebooks_Ultrabooks/ASUS_ZENBOOK_UX31A/#specifications [Accessed June 01, 2013].

[8] Guitar Instructions, 2012. G Major Chord.[Image Online] Available at: http://guitarinstructions4u.com/wp-content/uploads/2010/08/OpenG.jpeg [Accessed March 27, 2013]

[9] GuitarLessons.com 2012. Finger Numbers [Image Online] Available at: http://www.guitarlessons.com/media/guitar-lessons/chord-charts/finger-numbers.gif

[10] Wikipedia contributors, 2013, Arpeggio. Wikipedia, The Free Encyclopedia. Available at: http://en.wikipedia.org/wiki/Arpeggio [Accessed May 15, 2013]

[11] Wikipedia contributors, 2013, Articulation. Wikipedia, The Free Encyclopedia. Available at: http://en.wikipedia.org/wiki/Articulation_(music) [Accessed May 1, 2013]

[12] Wikipedia contributors, 2013, Dynamic Programming. Wikipedia, The Free Encyclopedia. Available at: http://en.wikipedia.org/wiki/Dynamic_programming [Accessed May 15, 2013]

[13] Wikipedia contributors, 2013, Exponential Growth. Wikipedia, The Free Encyclopedia. Available at: http://en.wikipedia.org/wiki/Exponential_growth [Accessed May 16, 2013]

[14] Wikipedia contributors, 2013, Hueristic(Computer Science). Wikipedia, The Free Encyclopedia. Available at: http://en.wikipedia.org/wiki/Heuristic_(computer_science) [Accessed May 15, 2013]

[15] Wikipedia contributors, 2013. Note. Wikipedia, The Free Encyclopedia. Available at: http://en.wikipedia.org/wiki/Note [Accessed April 13, 2013].

[16] 40662n3055, 2009. For The Love of God lesson Part 1. [video online] Available at: http://www.youtube.com/watch?v=L2I22QmNhz8 [Accessed April 11, 2013]

[17] docjs314, 2008. Steve Vai - Jibboom. [video online] Available at: http://www.youtube.com/watch?v=m6qf9RQhjqs [Accessed April 11, 2013]

[18] GuitarLessons365Song, 2011. AC/DC - Thunderstruck Guitar Lesson Pt.1 - Intro. [video online] Available at: http://www.youtube.com/watch?v=fkk2TlRc428 [Accessed April 11, 2013]

[19] JustinSandercoe, 2009. Minor Pentatonic Scale (Guitar Lesson BC-176) Guitar for beginners Stage 7. [video online] Available at: http://www.youtube.com/watch?v=G-X1RemAzks [Accessed 3 May, 2013]

[20] jun626, 2009. The Ocean. [video online] Available at: http://www.youtube.com/watch?v=LgplgyFrIGs [Accessed April 10, 2013]

[21] MissMisstreater, 2009. Guthrie Govan - Uncle Skunk Lesson - Levi Clay. [video online] Available at: https://www.youtube.com/watch?v=Q7JBD_F9WJ0 [Accessed 11 April, 2013]

[22] RoboCop00, 2006. Sweep Picking Practice. [video online] Available at: http://www.youtube.com/watch?v=3mbx03mP5eg [Accessed 11 April, 2013]

[23] r1ku97, 2011. Paul Gilbert - Technical Difficulties (Racer X). [video online] Available at: http://www.youtube.com/watch?v=rn-wj4pRpIE [Accessed April 10, 2013]

[24] tokairock, 2009. Led Zeppelin - Black Dog Live (TSRTS) by jun626. [video online] Available at: http://www.youtube.com/watch?v=gXvQfoa_qlw [Accessed April 10, 2013]