Abstract

Almost every note on the guitar can be played in several different locations, using one of several fingers. The number of ways a musical piece can be played therefore grows exponentially with the number of notes in the piece. Several things influence a guitar player’s choice of fingering. This study focuses on the biomechanical difficulty, using dynamic programming and a rule-based cost function to find the fingering that is easiest to play. The rules were mostly proposed by Mårten Falk, a guitar teacher at Lilla Akademien. The algorithm is limited to monophonic music. It was evaluated by Anders Eriksson, a guitar teacher at Lidingö Music School, on five classical music pieces. The produced fingerings were fully playable, though a few shortcomings were present, mostly stemming from the algorithm’s short-sightedness, where only adjacent notes are considered by the cost function.

Sammanfattning

På en gitarr får man oftast välja var, och med vilket finger, en not ska spelas. Ett musikstycke kan därför spelas på ett antal sätt som växer exponentiellt med antalet noter i stycket. Vilken fingersättning som är bäst beror på flera olika faktorer. Denna studie fokuserar på den biomekaniska svårigheten för en fingersättning. Den föreslagna algoritmen använder dynamisk programmering och en regelbaserad kostnadsfunktion för att hitta den mest lättspelade fingersättningen. Den är begränsad till monofonisk musik. De flesta reglerna föreslogs av Mårten Falk, gitarrlärare vid Lilla Akademien. Algoritmen kördes på fem klassiska musikstycken och resultaten utvärderades av Anders Eriksson, gitarrlärare vid Lidingö Musikskola. De genererade fingersättningarna var fullt spelbara. Det fanns dock några tillkortakommanden i dem, varav de flesta härstammade från algoritmens kortsiktighet, där enbart intilliggande noter påverkar kostnadsfunktionen.

Statement of Collaboration

This study was jointly performed by Vladimir Grozman and Christopher Norman. Christopher created the structure and framework of the algorithm, while Vladimir created the MIDI input module. The rules were written and weighted in collaboration. In this report, Vladimir mostly wrote *Sammanfattning/Abstract, Introduction, Cost Function,* and the Appendix. Christopher chiefly wrote the *Method, Results* and *Discussion*. The final text was, however, redacted to a great extent by both authors.

Table of Contents

Introduction

Some musical instruments have a one-to-one correlation between a note and a position on the instrument, such as a key or a tone hole, while others present several different locations where a note can be played. The same thing applies to fingering: for some instruments, one such position is always controlled by the same finger etc., while other instruments require a conscious decision by the player. Guitar playing combines these two problems.

Musical compositions written for the guitar can be played with a staggering number of different fingerings. Guitarists generally use rules and heuristics drawn from their experience to find reasonable fingerings, implying a certain degree of subjectivity. The set of rules may even change from one piece to another, depending on the genre of music and the guitarist's interpretation. [1]

Some principles are more universal than others, however, and while it may be impossible to automatically generate the best fingering for a given piece, important aspects of the problem are certainly approachable. If we focus on the biomechanical difficulty of playing a certain fingering, we can develop a set of rules that assign a cost to different elements of the fingering. The problem can then be seen as a graph problem, where the path with the lowest cost corresponds to a fingering with the least possible difficulty for the player. The research in this area is scarce, however, and the currently best available algorithms still produce fingerings subpar to those produced by professional guitarists.

This report is primarily aimed towards people with a good understanding of algorithms and complexity, e.g. computer science students. A musical background is not required, as the musical concepts and terms used are explained in the Appendix.

Scope

The aim of this study was to implement and evaluate an algorithm that proposes a guitar fingering for a given sequence of notes. The design criteria included biomechanical constraints of the hand and fingers and a set of basic guitar playing principles, taking the duration of the notes into account. The input is a monophonic sequence of notes in MIDI format and the output is a guitar tablature, additionally containing fingerings and position changes. This is simply a more human-friendly representation of (position, finger, string, fret, duration)-tuples.

We ignored the difficulty of plucking different strings with the right hand in this study. We were also solely concerned with the ease-of-play aspect of the left-hand fingering, ignoring any musical consideration, like the different tone quality of same note played on different strings. Ease-of-play from the point of view of a novice was prioritized.

The study includes only an optimization approach to the problem with no ability to learn from professional fingerings, as described in the *Background*. The reason was the limited time available for the completion of the study and the lack of readily available fingerings made by guitarist in a machine-readable format. Further, the amount of evaluation performed was limited.

Background

A straightforward way to approach the fingering problem is the so-called expert system approach as outlined by Sayegh. [1] It is a top-down approach where rules are formulated that dictate how to proceed from one fingering to the next when searching for a complete fingering for a piece. The rules are allowed to be ambiguous, i.e. they may allow transitions from one fingering to multiple alternatives, in which case certain transitions may be given priority. When using depth first search with backtracking the algorithm will yield a path that gives priority to the early notes in the piece. The quality of the produced fingerings will be dependent on the quality of the rules, which are based on the intuition of the authors and/or interviews with professional guitarists.

The approach taken in this study is another top-down approach which is outlined by Radicioni and Lombardo, [2] henceforth called the optimization approach. In this approach, the problem is translated into a graph problem where one attempts to find the path with minimum cost through a graph representation of the problem. In the graph, the vertices correspond to different fingerings of individual notes, and the edges correspond to the transitions between the fingerings. The cost assigned to each edge is an estimation of the difficulty the transition would pose to a guitarist.
The difference compared to the expert system approach outlined above is that in the expert system the constraints are hard constraints, i.e. the constraints specify which transitions are possible, whereas in the optimization approach the constraints are soft. Also, unlike the expert system approach the algorithm will yield a path that is globally optimal in terms of the cost function. Like the rules in the expert system approach the cost function could be based on interviews with professional guitarists and/or the intuition of the authors. Finding the best fingering, i.e. the path through the graph that minimizes the sum of costs of the included edges can be done efficiently with e.g. dynamic programming.

An approach similar to the optimization approach is outlined by Radisavljevic and Driessen, [3] where the same method is used in the forward phase of the algorithm. Unlike the purely top-down approach outlined above however, the algorithm is trained on fingerings produced by professional guitarists. The current optimum path is calculated in a forward phase. The algorithm then does a backward phase, where the optimum path as produced by the algorithm is compared to the training data and the error measure is minimized using back propagation with gradient descent.
The authors also propose using a multilayer artificial neural network for approximating the cost function which has the side effect of giving the system the ability to generalize from the fingering transitions it has encountered.

Top-down approaches have the advantage that they don’t need training data sets. They also have the advantage that the parameters for the rules/cost function can be adjusted to produce fingerings that prefer e.g. changing the hand position over switching strings or prefer using certain fingers over others, to suit the tastes of the intended user. This, of course, has to be

done carefully, by hand, but when using a bottom-up approach the same task amounts to collecting a large body of fingerings produced by the guitarist one wishes to emulate.
The main problem when using bottom-up approaches, i.e. algorithms that learn from examples, in this case is that professional fingerings are not readily available in a machine-readable format, so the amount of training examples will most likely be severely limited. In one of the papers cited, [3] the authors only had 7 training examples, which, according to the authors, were too few to use any of them for testing. Since the authors apparently tested the performance of the algorithm on the training examples one might suspect the exceptionally good results presented to be a product of overfitting.

Method

Algorithm

The proposed algorithm creates a graph interpretation of the problem, where the vertices correspond to the possible fingerings of each note in the composition and the edges correspond to the transitions from the fingerings of one note to the fingerings of the next note with a corresponding cost (see Figure 1). The graph is thus composed of layers where the first layer corresponds to all possible fingerings of the first note, the second layer corresponds to all possible fingerings of the second note, and so on. Since all transitions from each fingering in one layer to each fingering in the next layer are possible, the graph is completely connected between each layer.

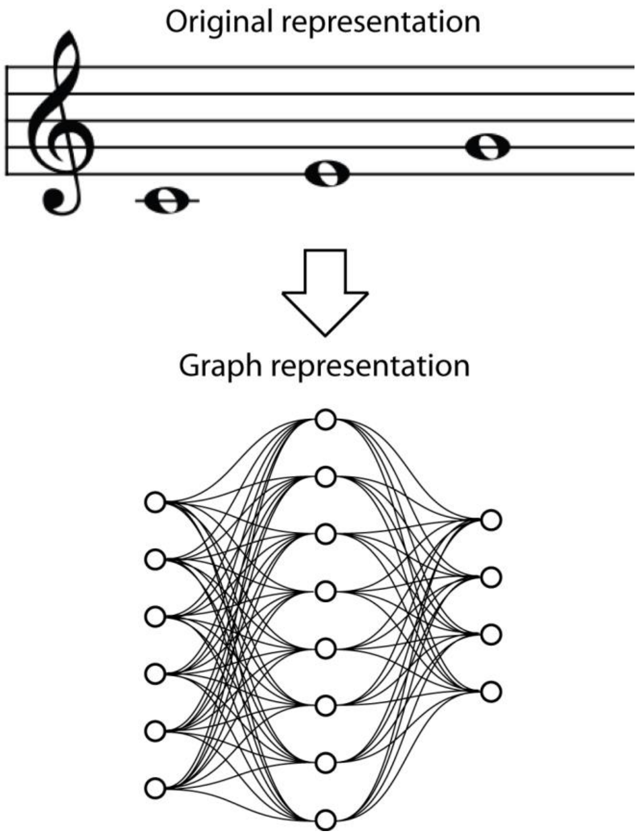

Figure 1. A schematic representation of the graph representation used by the algorithm. Each circle (node) in the graph representation corresponds to a single fingering, and each column of nodes corresponds to the note directly above it. The figure is merely intended to outline the structure of the graph representation so each note shown does not necessarily generate the number of nodes implied by this graph.

The cost function used in dynamic programming is local in nature, meaning that the cost is only calculated between nodes in successive layers in the graph. This must be taken into consideration when designing the cost function.

Each node can be specified as a tuple consisting of (1) the string number, (2) the fret number, (3) the finger number (if any), (4) the note duration, and (5) the hand position, i.e. the position of the index finger along the fretboard.
Any note corresponds to several potential fingerings. For example, with a standard tuning, a treble A (A_4 = 440 Hz) can be played at fret 5 on string 1, at fret 9 on string 2, or at fret 14 on string 3, and possibly, if the shape of the guitar allows the player to reach it, at fret 19 on string 4. Each of these can in turn use any of the possible four fingers on the left hand. In this algorithm only frets up to 14 are considered, but any treble A note will still produce 3 × 4 individual nodes.
Open notes (strings not pressed down) will produce a separate node for each hand position. Thus bass E (E_2 = 82.41 Hz), corresponding to the lowest open string, will produce 14 separate nodes in the algorithm.

The above graph approach might seem to produce an excessive amount of nodes. Might it not be possible to implement the algorithm without explicitly creating a node for each fingering of each note, however unlikely to appear in the optimal path? The answer is yes: the algorithm does not necessarily preclude the use of other search algorithms that give an optimal path, e.g. Dijkstra, in which the nodes might be generated lazily as they are encountered. The use of the above algorithm is chosen for simplicity, and for clarity of exposition, not for optimal runtime. The algorithm as written is, however, linear in the number of notes in the composition and the execution finished with no noticeable delay during testing, so changing the algorithm for the sake of optimization was deemed unnecessary. Also note that using e.g. Dijkstra would require using a priority queue which normally requires O(log *n*) time for insertions, thus resulting in an O(*n* log *n*) runtime.

Cost Function

The cost function uses a set of rules to determine the cost of any transition. These rules were, unless indicated otherwise, suggested by an experienced guitar player[1]. They were then manually weighed against each other in an attempt to produce the most reasonable fingerings.

Prefer lower frets

From a novice's point of view, playing in the first position (at the head) is greatly preferred. The higher up on the fretboard, the harder it becomes to play. A penalty is applied when not playing in the first position and another penalty is applied proportional to the current hand position.

It should be noted that players prefer positions close to the head of the guitar for reasons of familiarity rather than any increased biomechanical difficulty in using higher hand positions. Low and high positions on the fretboard are perceived as equally difficult by professional players, [4] but that is not the primary target group of this study. Penalizing higher frets is consistent with current guitar pedagogy as well as with pedagogical literature aimed at beginners, which encourage them to use hand positions close to the head of the guitar. [2] Even though this rule

[1] Mårten Falk, guitar teacher at Lilla Akademien, Stockholm. (2013-02-07)

is not based on biomechanical difficulty, it is fundamental and required to produce fingerings that resemble those produced by guitarists.

Avoid changing the hand position

Guitar playing is time-constrained. Large-scale, time consuming hand movements, like hand repositionings, are preferably avoided. [4] Thus, all changes of hand position incur a set penalty, as well as a penalty proportional to the number of frets the hand is moved. This applies to all hand position changes, but the penalty is reduced when the hand position changes during open notes, or when using a guide finger, i.e. when a finger is pressing a single string during the change in hand position.

Next finger on next fret

The natural finger placement for a given hand position is to place the index finger on the first fret in the position, the long finger on the second fret, the ring finger on the third fret, and the little finger on the fourth fret. When a finger is displaced from the natural position, a penalty is applied in order to favor placing each finger on a consecutive fret. Only the following placements are allowed:

- Index finger: always on the first fret.
- Long finger: always on the second fret.
- Ring finger: on the second, third or fourth fret.
- Little finger: on the second, third or fourth fret.

Stretching and cramping is additionally penalized by a different rule, the *Physical stretch/cramp penalty*.

Avoid using the same finger consecutively on two different strings

A finger can slide along a string, producing a smooth transition between two notes. Using the same finger consecutively on two *different* strings, however, is very inconvenient and should be avoided (unless a bar chord can be used). Thus doing so is heavily penalized.

Next finger on next string

This rule is similar to *Next finger on next fret*, except it is applied across the strings instead of along the fretboard. If, for instance, the index finger is placed on string 4 at a given fret, and the next note is placed on string 3 at the same fret, then it is natural to use the long finger rather than the ring or the little finger. Penalties are applied in proportion to the deviation from this natural finger placement. This rule does not apply to bar chords.

Finger penalty

Each finger incur a set penalty whenever it is used based on the perceived biomechanical constraints of the hand. The index and long fingers are stronger and more flexible than the ring and little fingers, and are therefore given priority.

Finger transition penalty

Similarly to individual finger penalties, penalties are applied individually to each possible finger transition. Transitions to and from open notes are free, as are self-transitions, i.e. not changing the finger. In particular, changing from the long finger to the ring finger and vice versa are penalized, as the movements of these fingers are biomechanically interdependent and thus more difficult to use consecutively.

Multiplicative penalty based on note length

During a long note, it is easier to move the next finger into a position where it can be quickly applied to the next note, or prepare for a change of hand position, when required. The total cost of a transition is therefore multiplied by a penalty factor based on the note length, e.g. making difficult transitions even more difficult when starting on a short note.

Physical stretch/cramp penalty

The algorithm includes a physical model for the distances between frets and strings on a guitar, as well as roughly measured “relaxed” and “cramped” positions of the fingers. Placing the fingers closer together than the cramped position, or further apart than the relaxed position, incurs a penalty that is proportional to a power of the displacement. This rule was adapted from Parncutt et al. [5]

Evaluation

The performance of the algorithm was evaluated on five classical pieces of varying difficulty. The specific pieces were partly chosen because they had little or no polyphony. Only the melody was used in the pieces which did include polyphony. An experienced guitarist[2] was then asked to play the pieces and mark the parts of the fingerings that he considered awkward, i.e. unlikely to be seriously considered by a professional guitarist. We then discussed these parts; if they could be considered wrong or merely non-optimal, whether they could be easily corrected manually, and what might have caused the problem.

The following pieces were chosen. The number of measures is provided as a reference.

1. J.S. Bach’s Minuet in G Major, arranged for guitar by Mårten Falk. 32 measures.
2. J. Haydn’s Symphony 94, the ‘Surprise Symphony’, arranged for guitar by Mårten Falk. 16 measures.
3. G.F. Händel’s Prelude, arranged for guitar by Eythor Thorlaksson. [6] 28 measures.
4. M. Carcassi’s Op. 11 nr 9 ‘Waltz’, arranged by Eythor Thorlaksson. [6] 32 measures.
5. Mauro Giuliani’s Op. 71 nr 1 ‘Sonatina’, arranged by Eythor Thorlaksson. [7] 72 measures.

[2] Anders Eriksson, a guitar teacher at Lidingö Music School, Stockholm. (2013-04-09)

Results

According to the professional guitarist, there were no obviously wrong or unplayable parts in any of the produced fingerings. There were several choices made by the algorithm, however, that he considered strange or non-optimal. These issues are listed in order below. Pieces 1 and 4, for which the algorithm produces no issues, are omitted.
In the excerpts, Arabic numerals denote the intended finger and Roman numerals denote hand positions. Fingerings are omitted when they are obvious to a guitarist, as is usual when indicating fingerings for sheet music. Any notes with implicit fingerings are explained in the figure captions.

Surprise Symphony

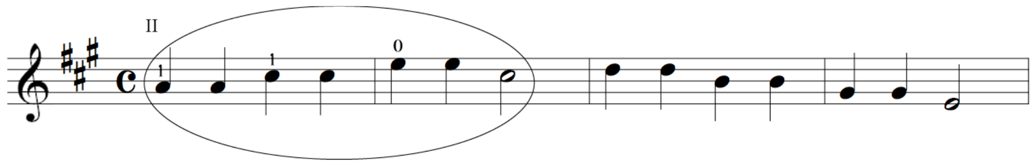

Figure 2. Excerpt from the fingering produced for J. Haydn’s Surprise Symphony, measures 1-4. The awkward part has been outlined. Only the first occurrence of each note has been marked with the intended finger: subsequent occurrences of the same note are supposed to use the same finger.

Issue 1: In Figure 2 the algorithm chooses to play the first four notes with a bar chord over the second fret which the player has to lift in order to play the fifth note open on the first string. A simpler way to play the same excerpt would be to use fingers 2, 2, 3, 3 for the first four notes, leaving the first string open (the fingering suggested by Mårten Falk in the original arrangement). Alternatively, one might use the fingering produced by the algorithm, but play the fifth and sixth notes on the second string at fret 5, which would obviate the need to release the bar chord (the fingering suggested by Anders Eriksson). This would also mean that one could use the bar chord to play the seventh note.

Prelude

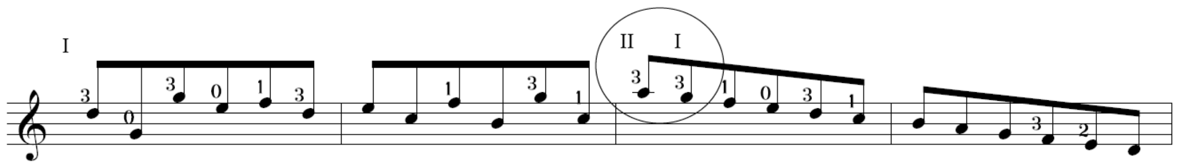

Figure 3. Excerpt from the fingering produced for G. F. Händel’s Prelude, measures 25-28. The original score has bass notes which have been removed to satisfy the monophony constraint of the algorithm. The awkward part has been outlined.

Issue 2: In Figure 3 the algorithm produces a hand position change from position 1 to position 2 in order to slide the ring finger from the third to the fifth fret. However, it would have made more sense to move the left hand to position 3 in order to reduce the hand stretch necessary to reach fret 5 with the ring finger.

Sonatina

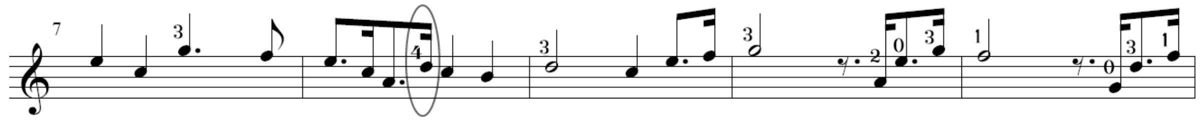

Figure 4. Excerpt from the fingering produced for Mauro Giuliani's Sonatina, measures 7-11. The original score has bass notes which have been removed to satisfy the monophony constraint of the algorithm. The awkward part has been outlined.

Issue 3: In some cases the algorithm's choice between the ring finger and the little finger could seem arbitrary, as in Figure 4, where the algorithm has chosen to play treble D with the little finger (outlined), but then chooses to play treble D with the ring finger in the third measure.

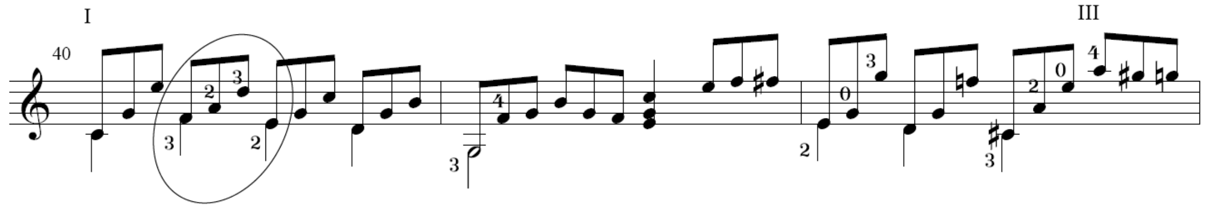

Figure 5. Excerpt from the fingering produced for Mauro Giuliani's Sonatina, measures 40-43. The original score has bass notes which have been removed to satisfy the monophony constraint of the algorithm. The three concurrent notes in measure three were separated into individual notes when the piece was entered into the algorithm. The awkward part has been outlined.

Issue 4: The outlined notes in Figure 5 are playable, but would have been easier to play using fingers 3, 2, 4, 1. Doing so would have required changing the hand position to position 2, however, because the algorithm does not allow moving the index finger from the first fret in a hand position.

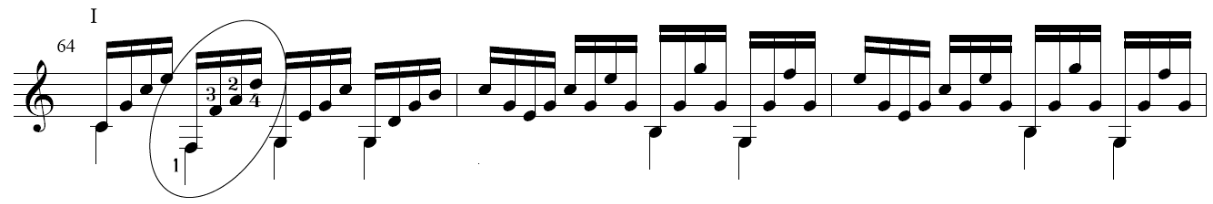

Figure 6. Excerpt from the fingering produced for Mauro Giuliani's Sonatina, measures 64-66. The bass notes have been kept in this case since they alternate with the melody. Compare the outlined part with the outlined part in Figure 5.

The outlined parts in Figures 5 and 6 offer an interesting comparison. The last three notes in the outlined part in Figure 6 are the same as the first three notes in the outlined part in Figure 5, yet in Figure 6 the algorithm has produced a fingering that uses all four available fingers on the left hand, which is preferable.

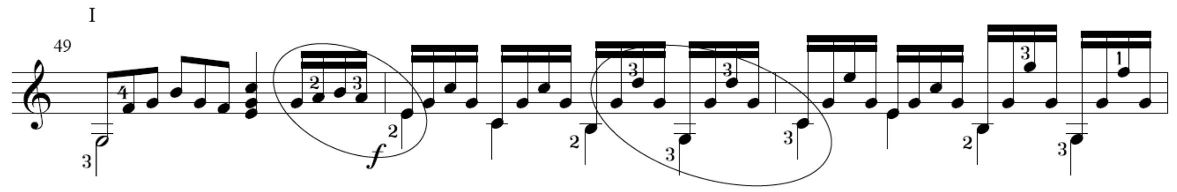

Figure 7. Excerpt from the fingering produced for Mauro Giuliani's Sonatina, measures 49-51. The bass notes have been kept in this case since they alternate with the melody. The three concurrent notes in the first measure were separated into individual notes when the piece was entered into the algorithm. The awkward parts have been outlined. The outlined notes for which the finger has not been marked are intended to be played open.

Issue 5: During the transition from the first measure to the second in Figure 7 (the first outline) the algorithm first switches from using finger 2 to finger 3 to play the second and fourth notes, and then uses finger 2 to play the fifth note. While this is perfectly playable, one might simply use the third finger on the second note, completely obviating the need to change fingers between the second and fourth notes.

Issue 6: Again, in Figure 7 in the second and last measure (the second outline), the algorithm uses finger 3 to play both the bass notes and the melody, i.e. the player has to move the third finger to alternatingly press string 2 and string 6. This would easily have been avoided by using fingers 3 and 4 instead. This problem occurs in several places in this piece.

Discussion

It is worth pointing out once again that the fingerings produced by the algorithm are almost entirely uncontentious – the issues are the exception, not the rule. Furthermore, the issues listed above were easily corrected by the experienced guitarist during the interview, with only minor modifications of the fingerings.

Issue (1) is likely caused by the parameters used in the cost function. Each of the first three transitions in the outlined section in Figure 2 incur no penalty at all and no special consideration is given to releasing bar chords, so the transition from the fourth to the fifth note in Figure 2 incurs no more penalty than if the third note had been fingered with the index finger, without using a bar chord. Note the difference between these cases: when using a bar chord one also presses the first string, which in the fingering in Figure 2 is played open for the fifth note. Normally when using the index finger one leaves the first string open, which means there is no need to move any fingers of left hand in order to play the fifth note.
To counteract issue (1) one might add a cost for releasing bar chords, but then one would need to add some way to mark if nodes are part of a bar chord, since at present the algorithm doesn't distinguish between using bar chords and normal use of the index finger.

Similarly, issue (2) is likely caused by the parameters used in the cost function and does not merit extensive mention.

Arbitrariness, as exemplified by issue (3), is a natural consequence of the locality of the cost function, and is not possible to avoid under the assumptions of this algorithm. Furthermore, this study concerns the biomechanical difficulty of the fingerings and does not consider consistency. This issue was however pointed out by the experienced guitarist and so is included for the sake of completion.

Issues (4), (5), (6) are examples of a recurring problem with this algorithm, and are all an expected consequence of the locality of the cost function. Since no nodes before the immediately preceding node are taken into account when calculating the cost, the algorithm frequently produces passages that are awkward when considered in context but where each pair of successive fingerings are perfectly reasonable when considered on their own. Since about two thirds of the Sonatina contain the recurring treble G which can (and is intended to) be played open, and since transitions to and from open notes incur a very low penalty, the performance of the algorithm is affected in several places.

Given the above, and given that arrangements like the Sonatina with an alternating open note are common in guitar music, one might consider modifying the algorithm to let each node correspond to fingerings of two successive notes instead of fingerings of individual notes. [5] Doing so would produce more nodes, but would likely have avoided issues (4), (5) and (6). Another possible way to address issues (4), (5) and (6) would be to let each node in the algorithm denote multiple fingers, i.e. chords, which would also eliminate issues of this kind in cases where there is more than one intermediate open note. As a side effect, the algorithm would also be able to handle polyphony. Another way of handling polyphony is discussed in, [2] where the cost function is applied between each pair of notes in a chord. Letting each node correspond to chords instead of individual fingerings would have been considerably more difficult to implement, however, lying well beyond the scope of this study.

Conclusion

The problem of guitar fingering is very interesting algorithmically: it is a problem of human behavior modeling with clearly defined boundaries. Unsurprisingly then, it has an array of interesting solutions, although none of them can match an experienced guitar player.

That human players outperform automatic fingering generation algorithms does not mean, however, that such algorithms cannot be useful. Several use cases can be presented. Music notation software is one example, where staff and tablature notation can be displayed side-by-side. They can also be of help to beginners, who might have a hard time reading staff notation.

In this study, we have implemented an algorithm for guitar fingering that works reasonably well within the defined limitations. In its current form, it is limited to monophonic music, which is of limited use in real life, but it could theoretically be modified to support polyphony. Changing the nodes to represent fingerings of two adjacent notes could also remedy some of the short-sightedness. Most of the issues were easily detected and corrected, but it still shows that the output of the algorithm cannot be accepted uncritically. It could however be used as a starting point, for instance in the use cases presented above, potentially saving a lot of time for both novices and professionals.

References

1. Sayegh S. Fingering for string instruments with the optimum path paradigm. Comput. Music J. 1989;13(3):76-84.
2. Radicioni D, Lombardo V. Guitar fingering for music performance. Proceedings of the International Computer Music Conference; 2005 Sep 5-9; Barcelona, Spain.
3. Radisavljevic A, Driessen P. Path difference learning for guitar fingering problem. Proceedings of the International Computer Music Conference; 2004 Nov 1-6; Miami, USA.
4. Heijink H, Meulenbroek RGJ. On the complexity of classical guitar playing: functional adaptations to task constraints. J. Motor Behav. 2002;34(4):339-351.
5. Parncutt R, Sloboda J, Clarke E, Raekallio M, Desain P. An ergonomic model of keyboard fingering for melodic fragments. Music Percept. 1997;14(4):341-382.
6. Thorlaksson E. [1033] Guitar Moment II. No date [cited 2013 Apr 12]. Available from: http://www.classicalguitarschool.net/en/Download.aspx?id=1033
7. Thorlaksson E. [1086] Guitar Moment IV. No date [cited 2013 Apr 12]. Available from: http://www.classicalguitarschool.net/en/Download.aspx?id=1086

Appendix. Some basics of music and guitar playing.

Most people are probably familiar with some musical concepts through the example of a piano or a keyboard (see Figure A-1). When pressed, each of the keys emit a sound with a different pitch, a *note*. Besides pitch, the duration of a note is another important characteristic. Notes are usually written in staff (or standard) notation, the details of which shall not be discussed here. It is enough to know what notes and *measures* look like, see Figure A-2.

Figure A-1. Keys on a keyboard, each representing a different note.

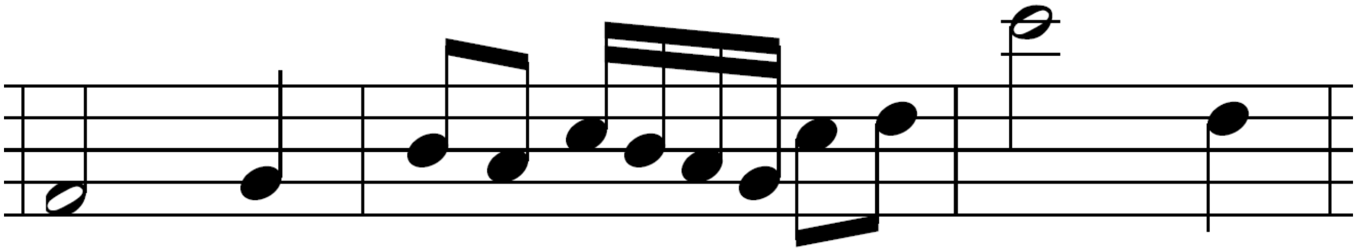

Figure A-2. Twelve different notes. The four vertical bars divide the notes into measures. This piece of music is monophonic, since only one note is played at a time. In polyphonic music, notes are stacked vertically, as seen in Figure 7.

Things are more complicated on a guitar. Sound is produced by vibrating strings, but there are only six of them on a standard guitar. In order to play more than six different notes, the vibrating part of a string is shortened by pressing it at the appropriate location with a finger, increasing the pitch (see Figure A-3).

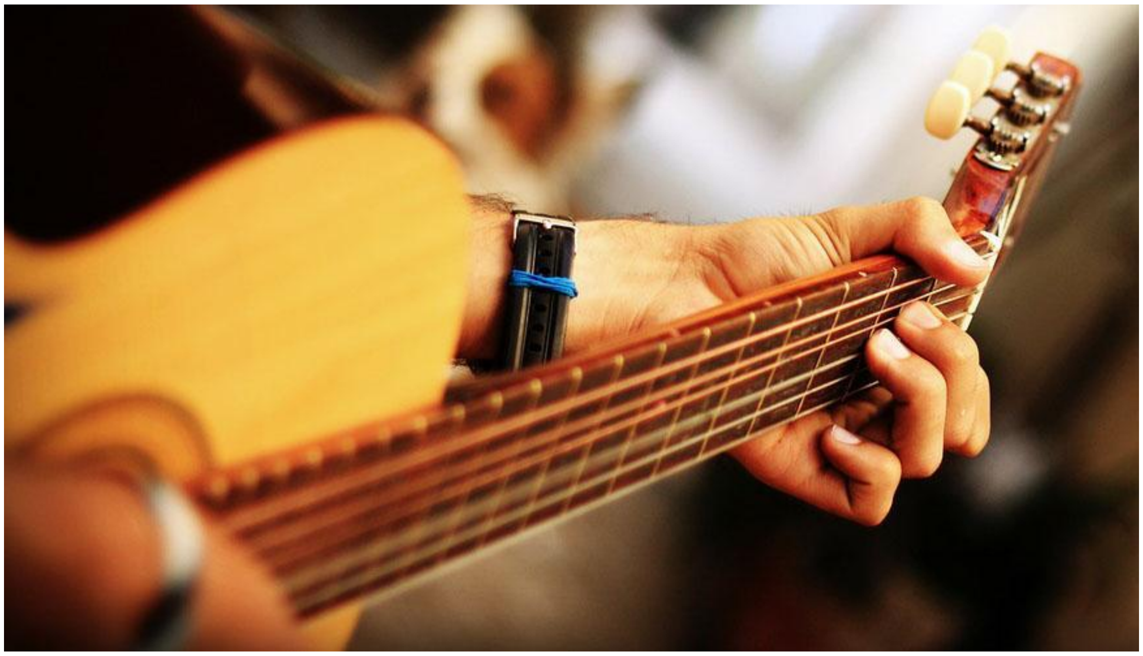
Figure A-3. Pressing two strings on the second fret.

These locations are called *frets*. The frets are the vertical bars that cross the *fretboard* (see Figure A-3). The fretboard is the front part of the *neck*, which runs between the *head* and the *body* of the guitar. The frets correspond to different notes, similarly to the keys on the piano, and up to six notes can therefore be played simultaneously. However, not every combination of a fret and a string produces a unique note, as there is an overlap between the strings. With some exceptions, a note that is played on one string can also be played on the next string (upwards), five frets further up (towards the body).

For both the piano and the guitar, each note can be played/pressed using one of several fingers: ten fingers on the piano and four or five on the guitar, depending on the type of guitar. A note on the guitar can also be *open*, which means that the string is not pressed and vibrates at full length. Thus, we are back at the original problem presented in the *Introduction*: when playing a piece of music, which is basically a list of notes, on what string and fret should they be played, and using which finger?

Some more terminology is required to properly articulate a solution. When playing, the player's hand can move up and down the fretboard (which is actually left and right) in order to reach all frets. This results in a change of *hand position*. There is a link between positions and frets: when the hand is in position *X*, it means that the fingers are located so that the index finger can comfortably press a string on fret X, the middle finger on fret X + 1, the ring finger on fret X + 2 and the little finger on fret X + 3 (see also the *Next finger on next fret* rule). These fingers are designated *finger 1* through *4*.

Recommended fingerings can be added to the staff notation of a note by placing the corresponding Arabic digit next to it. This is generally omitted when the fingering is implicit.

Hand positions are commonly denoted by Roman numerals. This is the format used in the *Results*.

Usually, one finger is used to press one string, but a finger can also be placed vertically across several strings. This is called a *bar chord* and is normally only performed with the first finger.

A polyphonic piece (see Figure A-2) can sometimes be divided into a melody and a bass part, which are largely monophonic in the pieces chosen in this study. These roughly correspond to the right and the left hand of a pianist, see Figure A-1.