HIGH SCHOOL EDITION
Journal of Student Research
Volume 12 Issue 4 (2023)

# Optimizing Fingerings on Bowed Instruments *(particularly the cello)*

*My fingers have to figure out where I need to be in order to get the right sounds out." - Yo-Yo Ma*

Jasper Lee

Cupertino High School

## ABSTRACT

While papers on using dynamic programming to generate fingering options for piano and guitar music have been published, there is a paucity of such exploration for bowed instruments. This paper adapts the technique of dynamic programming in an attempt to solve the challenge of automating the process of generating fingering options for cello music. By capturing the physical constraints faced by a beginner cellist with carefully-designed transition functions, this paper demonstrates how the algorithm can generate sensible fingering options for cello passages. The novelty of this paper lies in the fact that one such possibility can be explored with the violoncello, thus broadening the application of computational science in the performing arts.

## Introduction

In the 1950s, American applied mathematician Richard Bellman developed a method for mathematical and computational programming optimizations known as 'dynamic programming'. His creation not only broke down extensive sets of equations into smaller subsets but was able to determine the best or shortest solution to a given issue. As a cellist, one of the most notorious aspects of playing a piece is finding the most efficient fingering. Quite rarely does the very first fingering I think of become the fingering that I will end up using, as it requires a certain number of repetitions at the "concert tempo" to serve as a reassurance that the fingering works–or an indication that it doesn't. Oftentimes, sheet music comes with a set of fingerings written by a master cellist, and is considered a "certified or safe" fingering, however, they are most of the time customized to fit personal styles or interpretations of the piece, and aren't necessarily the most "efficient." This presents students with unnecessary additional hurdles to overcome. In light of this, the dynamic programming algorithm presented in this paper is a potential resolution to that issue.

## Literature Review

### Finding Optimal Piano Fingerings

A standing approach has been made by Dr. Susan Kelly in her paper *Finding Optimal Piano Fingerings* (2011; UMAP), where she created four tables of difficulties (see Figure 2) categorizing the various types of motions one goes through while playing the piano and its respective difficulties ranging from 1 (easy) to 4 (hard). *Lower white-upper white basic movements* documents movement from one white key the next in the right direction; *Lower white-upper black basic movements* documents movement from one white key to a proceeding black key in the right direction;

| Fingers (lower,upper) | Interval size in half-steps 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| (1,2) | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 3 |
| (1,3) | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 3 |
| (1,4) | 2 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 |
| (1,5) | 3 | 3 | 2 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| (2,1) | 2 | 2 | 3 | 3 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| (2,3) | 1 | 1 | 1 | 1 | 2 | 2 | 3 | 3 | 3 | 3 | 3 | 3 |
| (2,4) | 2 | 2 | 1 | 1 | 1 | 1 | 2 | 3 | 3 | 3 | 3 | 3 |
| (2,5) | 3 | 3 | 2 | 2 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 2 |
| (3,1) | 2 | 2 | 3 | 3 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| (3,4) | 1 | 1 | 2 | 2 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| (3,5) | 3 | 3 | 1 | 1 | 1 | 1 | 3 | 3 | 3 | 3 | 3 | 3 |
| (4,1) | 2 | 2 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| (4,5) | 1 | 1 | 1 | 1 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| (5,1) | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |

**Table 2.**
Lower white–upper black basic movements.

| Fingers (lower,upper) | Interval size in half-steps 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| (1,2) | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 3 | • |
| (1,3) | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | • |
| (1,4) | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | • |
| (1,5) | 3 | 3 | 3 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 1 | • |
| (2,1) | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | • |
| (2,3) | 1 | 1 | 1 | 2 | 2 | 3 | 3 | 3 | 3 | 3 | 3 | • |
| (2,4) | 2 | 1 | 1 | 1 | 1 | 2 | 2 | 3 | 3 | 3 | 3 | • |
| (2,5) | 3 | 2 | 2 | 2 | 2 | 1 | 1 | 1 | 2 | 2 | 3 | • |
| (3,1) | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | • |
| (3,4) | 1 | 1 | 1 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | • |
| (3,5) | 3 | 2 | 2 | 2 | 2 | 2 | 3 | 3 | 3 | 3 | 3 | • |
| (4,1) | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | • |
| (4,5) | 2 | 2 | 2 | 2 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | • |
| (5,1) | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | • |

**Table 3.**
Lower black–upper white basic movements.

| Fingers (lower,upper) | Interval size in half-steps 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| (1,2) | 3 | 2 | 2 | 1 | 1 | 2 | 2 | 2 | 3 | 3 | 3 | • |
| (1,3) | 3 | 2 | 2 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 3 | • |
| (1,4) | 3 | 3 | 3 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | • |
| (1,5) | 3 | 3 | 3 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 1 | • |
| (2,1) | 2 | 3 | 3 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | • |
| (2,3) | 1 | 1 | 1 | 2 | 2 | 3 | 3 | 3 | 3 | 3 | 3 | • |
| (2,4) | 2 | 1 | 1 | 1 | 1 | 2 | 3 | 3 | 3 | 3 | 3 | • |
| (2,5) | 3 | 2 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | • |
| (3,1) | 2 | 3 | 3 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | • |
| (3,4) | 1 | 1 | 1 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | • |
| (3,5) | 2 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 3 | 3 | 3 | • |
| (4,1) | 3 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | • |
| (4,5) | 1 | 1 | 1 | 2 | 2 | 3 | 3 | 3 | 3 | 3 | 3 | • |
| (5,1) | 3 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | • |

**Table 4.**
Lower black–upper black basic movements.

| Fingers (lower,upper) | Interval size in half-steps 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| (1,2) | • | 2 | 2 | 2 | 2 | • | 3 | 3 | 3 | 3 | • | 3 |
| (1,3) | • | 2 | 2 | 2 | 2 | • | 2 | 2 | 2 | 2 | • | 2 |
| (1,4) | • | 3 | 2 | 2 | 2 | • | 2 | 1 | 1 | 1 | • | 2 |
| (1,5) | • | 3 | 3 | 3 | 3 | • | 2 | 1 | 1 | 1 | • | 1 |
| (2,1) | • | 2 | 3 | 3 | 4 | • | 4 | 4 | 4 | 4 | • | 4 |
| (2,3) | • | 1 | 1 | 1 | 2 | • | 3 | 3 | 3 | 3 | • | 3 |
| (2,4) | • | 2 | 1 | 1 | 1 | • | 2 | 3 | 3 | 3 | • | 3 |
| (2,5) | • | 3 | 2 | 2 | 1 | • | 1 | 1 | 1 | 2 | • | 2 |
| (3,1) | • | 3 | 4 | 4 | 4 | • | 4 | 4 | 4 | 4 | • | 4 |
| (3,4) | • | 1 | 1 | 1 | 2 | • | 3 | 3 | 3 | 3 | • | 3 |
| (3,5) | • | 3 | 1 | 1 | 2 | • | 3 | 3 | 3 | 3 | • | 3 |
| (4,1) | • | 4 | 4 | 4 | 4 | • | 4 | 4 | 4 | 4 | • | 4 |
| (4,5) | • | 2 | 2 | 2 | 3 | • | 3 | 3 | 3 | 3 | • | 3 |
| (5,1) | • | 4 | 4 | 4 | 4 | • | 4 | 4 | 4 | 4 | • | 4 |

*Lower black-upper white basic movements* documents movement from a black key to a white key in the left direction; *Lower black-upper black basic movements* documents movement a black key to a proceeding black key. (See

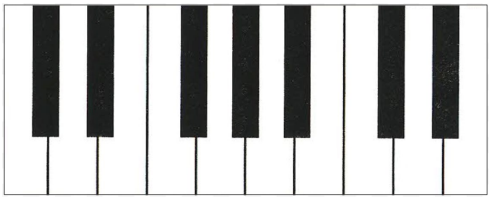


Figure 1 for a visual)

**Figure 1.** Simple visual representation of the Piano detailing the white and black keys.

**Figure 2.** Tabular documentation of the individual 'costs' in playing the 'next note' in a musical piece. The cost is determined by the current finger being used ('lower'), the next finger to be used ('upper') and the interval size (or note distance) the player has to cover when moving towards the right. The 'Fingers (lower, upper) 'column represents all the possible fingering combinations for one hand, and the 'Interval size in half-steps' row represents the number of half steps to each note. A half step is the relationship between a white key and a black key or consecutive white keys between two series of black keys (see Figure 1). Similarly, the 'Interval' is an accumulation of 'half-steps' documenting har far away the next note is.

As shown and described in Figure 2, Dr. Kelly's tables are based off of 3 distinctive components, the current finger, the next finger and the distance to the next note, and she assigns a 'cost' value to each possible combination of elements. Her algorithm then parses through a set of notes from a piece, iterates through all the possible fingerings, and computes the possible permutations and its respective difficulties, and arrives at the next best finger to put into the current set. This process continues until the end of the piece and returns with the optimal fingering.

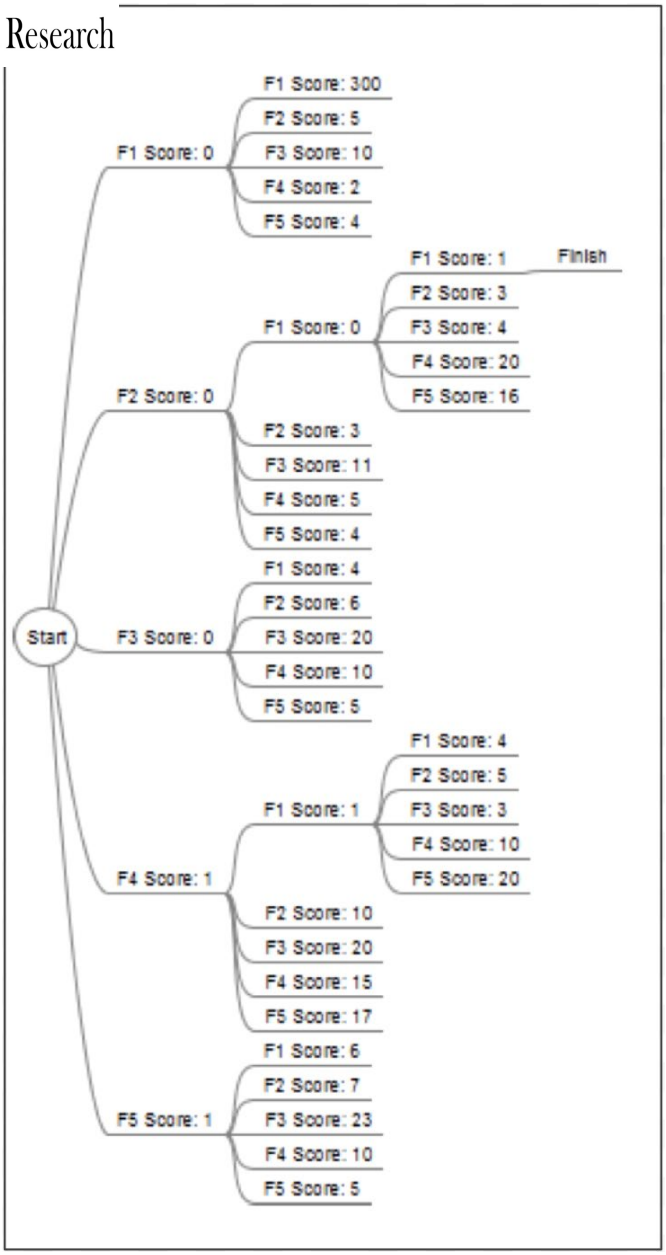
## Optimal Piano Fingerings With Comfortability

In a more recent study, Xingye Lu's *Optimal Piano Fingering for Simple Melodies* (2014; Royal Institute of Technology) has explored an alternative approach to the problem, utilizing comfortability in the hands. Lu categorizes the distances between each note given possible fingering pairs in a" reach table" (see Figure 3).

| Finger Pairs | MinPrac | MinComf | MinRel | MaxRel | MaxComf | MaxPrac |
|---|---|---|---|---|---|---|
| 1-2 | -5 | -3 | 1 | 5 | 8 | 10 |
| 1-3 | -4 | -2 | 3 | 7 | 10 | 12 |
| 1-4 | -3 | -1 | 5 | 9 | 12 | 14 |
| 1-5 | -1 | 1 | 7 | 10 | 13 | 15 |
| 2-3 | 1 | 1 | 1 | 2 | 3 | 5 |
| 2-4 | 1 | 1 | 3 | 4 | 5 | 7 |
| 2-5 | 2 | 2 | 5 | 6 | 8 | 10 |
| 3-4 | 1 | 1 | 1 | 2 | 2 | 4 |
| 3-5 | 1 | 1 | 3 | 4 | 5 | 7 |
| 4-5 | 1 | 1 | 1 | 2 | 3 | 5 |

**Figure 3.** Tabular documentation and categorization of the various degrees of 'reaching' with different finger pairs or fingerings. In the 'Finger Pairs' column, the numbers represent possible fingerings, where as in all other columns the numbers below represent the reach in terms of half steps. The largest and smallest spans used in practice sessions (MaxPrac & MinPrac) gauge the difficulty in performing a certain interval. The largest and smallest comfortables spans (Max Come & MinComf) describe the comfortability in the hands. The spans (MaxRel & MinRel) depict the maximum and minimum reach a finger pair can reach in a relaxed state.

With this table (see Figure 3), Lu is not only able to customize values according to a musician's hands shape and size, but is able to determine an optimized and personal fingering. The process is quite similar to that of Dr. Kelly's and involves sequential steps that optimize the "difficulty" or "points" –according to Lu–accumulated throughout a piece. Using different rules Lu references his reach table and assigns cost points accordingly. For instance Lu employs the 'Small-Span Rule' and assigns 1 point for each interval that is less than 'MinRel' or in the case of the 'Position-Change-Count Rule', 2 points are assigned whenever an interval is greater than MaxComf or less than MinComf. Similarly, Lu implements other rules such as the "Four-on-Black Rule" and the "Three-Four-Five Rule", in which 1




point is assigned, along with the "Thumb-on-Black Rule", where a total of 2 points could be assigned. These points are then added up to the total 'score 'of a given permutation, and passed into a comparison function that finds the optimal score (in this case the score with the least value). This algorithm constitutes a "Finger Tree", as shown below, and again takes on an approach similar to that of Dr. Kelly.

**Figure 4.** Visual representation of the 'Finger Tree' utilized by Lu. Every 'branch' of the tree represents a fingering permutation and is assigned a 'score' measuring its efficiency. After comparing the individual scores the algorithm arrives at the optimal fingering labeled 'finish.'

## Guitar Fingering

On MIT OpenCourseWare, Instructor Erik Demaine lectures on more of the same with the guitar. Despite the distinct characteristics of both instruments, they share the common ground of playing every note with a different finger. Where the piano has keys, the guitar has frets, therefore the same concept scan be applied. Demaine describes the process with two sub-functions: the 'min function 'that finds the minimum of difficulties and the 'cost function 'that gives us

the difficulty or cost of a certain permutation of fingerings. The ‘min function ’sequentially adds the costs that the ‘cost function ’generates for us given the current note and finger we are on and determines the next best possible finger. These two functions will continue with this process until the end of a piece and return us with the optimized fingering permutation.

### Implications

As evidenced, the implications that a pianist or guitarist faces for fingerings are quite complex and involve a variety of tables and values documenting aspects from the ‘cost ’of finger movement to the comfortability of a fingering, however their layers of complexity remain in the realm of invariably playing the next note with a different finger. For the cello on the other hand–and in that sense all other bowed strings instruments–we have to factor in sliding, shifting, and string crossings, which can be all be performed within the ability of one finger, on top of comfortability and cost, involving another set of intricacies to conquer.

## Assumptions and Notation

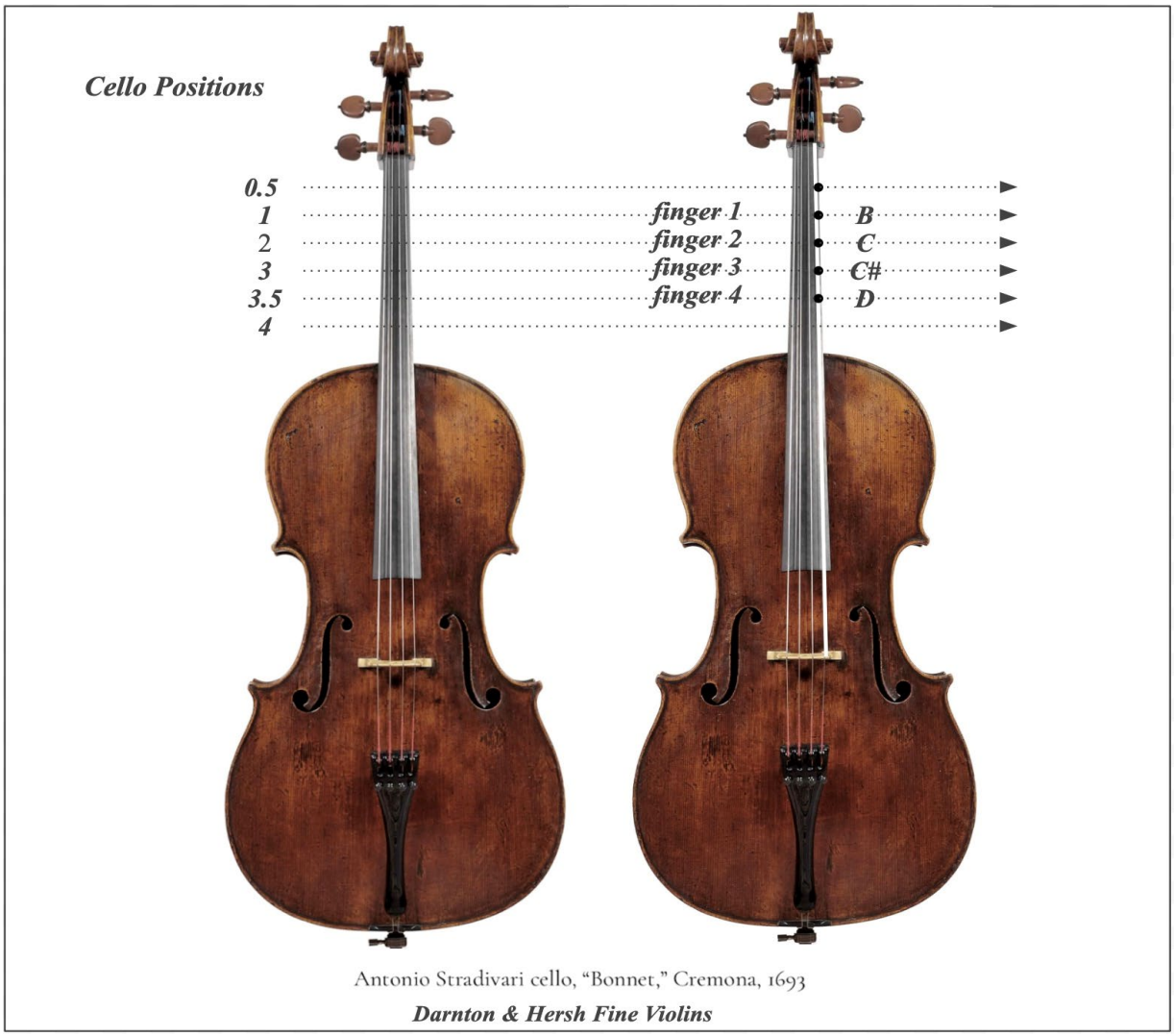


**Figure 5**. Diagram illustrating the 6 positions on the cello (0.5, 1, 2, 3, 3.5, & 4 positions) and their corresponding notes and fingers on the A string (string on the far right highlighted in white).

When playing the cello, or any other bowed string instruments, we come across two main types of movements: vertical and horizontal. For horizontal movements, we can categorize them into 'positions'. Although technically speaking, there are more than 30 possible positions on the cello, it has conventionally been counted up to the 4th position, starting with Half Position and incrementing by 0.5, leaving us with 6 Positions: 0.5 Position, 1st Position, 2nd Position, 2.5 Position, 3rd Position, 3.5 Position and 4th Position. Each position is then determined by the placement of the index finger, which in turn sets what notes the other fingers are on in successive half-steps. For instance on the A-string in 1st position, the index finger (1st finger) rests on B, the 2nd finger on C, the 3rd finger of C#, and the 4th finger on D (see Figure 5).

As for vertical movements, we encounter shifting (moving the left hand from one position to the next) which involves two components: 'shifting up 'and 'shifting down'. With each component we have 16 different possible finger combinations for each shift:

| | | | |
|---|---|---|---|
| *(1,1)* | *(2,1)* | *(3,1)* | *(4,1)* |
| *(1,2)* | *(2,2)* | *(3,2)* | *(4,2)* |
| *(1,3)* | *(2,3)* | *(3,3)* | *(4,3)* |
| *(1,4)* | *(2,4)* | *(3,4)* | *(4,4)* |

**Figure 6.** Matrix documenting the 16 possible fingerings on the cello. The digit towards the left of the parentheses represents the current finger being used and the other the next finger to be used.

This presents us with a measure of difficulty (the "cost") for each. Now this cost not only depends on the finger combination but the distance of shifting, which is measured in half steps. Given that we are ending at the 4th position, our tables for shifting up and shifting down will be quite different. During shifting, we move our hands from a lower position to a higher position or vice versa; depending on the current position we are in, the number of half steps differs, as the stopping point is at 4th position. Below is an example of a table documenting the 'costs 'for shifting up 1st position.

***1st Position***

**# of Half-Steps**

| Fingers (start, end) | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| **(1,1)** | *1* | *2* | *3* | *3* | *2* | *4* | *4* | *4* | *4* |
| **(1,2)** | *0* | *0* | *3* | *3* | *3* | *2* | *4* | *4* | *4* |
| **(1,3)** | *0* | *0* | *0* | *3* | *3* | *3* | *2* | *4* | *4* |
| **(1,4)** | *0* | *0* | *0* | *0* | *3* | *3* | *3* | *2* | *3* |
| **(2,1)** | *2* | *3* | *3* | *2* | *4* | *4* | *4* | *4* | |
| **(2,2)** | *1* | *3* | *3* | *3* | *2* | *4* | *4* | *4* | |
| **(2,3)** | *0* | *2* | *3* | *3* | *3* | *2* | *4* | *4* | |
| **(2,4)** | *0* | *0* | *2* | *3* | *3* | *3* | *2* | *3* | |
| **(3,1)** | *3* | *3* | *2* | *4* | *4* | *4* | *4* | | |
| **(3,2)** | *3* | *3* | *3* | *2* | *4* | *4* | *4* | | |
| **(3,3)** | *1* | *3* | *3* | *3* | *2* | *4* | *4* | | |
| **(3,4)** | *0* | *2* | *3* | *3* | *3* | *2* | *3* | | |
| **(4,1)** | *3* | *2* | *4* | *4* | *4* | *4* | | | |
| **(4,2)** | *3* | *3* | *2* | *4* | *4* | *4* | | | |
| **(4,3)** | *3* | *3* | *3* | *2* | *4* | *4* | | | |
| **(4,4)** | *1* | *3* | *3* | *3* | *2* | *3* | | | |

**Figure 7**. Table containing the individual costs of playing a note a certain number of half steps away with a pair of fingers in 1st Position. The column on the far left under "Fingers (start, end)" contains the current finger we have pressed down and the next finger that could be pressed, whereas under rows, we have the "# or Number of Half-Steps", which documents how far away the next note is. The data values are the 'costs 'that come with the pair of fingers, ranging from 0 (being easy) to 4 (being hard).

With each increment in the starting finger, there is a decrement in the number of half steps columns because the distance to 4th position decreases. On the other hand, for shifting down, we have similar tables. Where the shifting up tables stop at 4th position, the shifting down tables stop at 0.5 Position, the lowest possible position on the cello. Figure 8 is an example of a cost table documenting shifting down from 3.5 Position.

| 3.5 Position | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | # of Half-Steps | | | | | | | |
| **Fingers (start, end)** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
| **(1,1)** | *1* | *2* | *3* | *2* | *4* | | | |
| **(1,2)** | *2* | *3* | *2* | *4* | *10* | | | |
| **(1,3)** | *3* | *2* | *4* | *10* | *10* | | | |
| **(1,4)** | *2* | *4* | *10* | *10* | *10* | | | |
| | *G* | *F#* | *F* | *E* | *Eb* | | | |
| **(2,1)** | *0* | *1* | *2* | *3* | 2 | 3 | | |
| **(2,2)** | *1* | *2* | *3* | *2* | 3 | *10* | | |
| **(2,3)** | *2* | *3* | *2* | *3* | *10* | *10* | | |
| **(2,4)** | *3* | *2* | *3* | *10* | *10* | *10* | | |
| | *G#* | *G* | *F#* | *F* | *E* | *Eb* | | |
| **(3,1)** | *10* | *0* | *1* | *2* | 3 | 2 | 3 | |
| **(3,2)** | *0* | *1* | *2* | *3* | 2 | 3 | *10* | |
| **(3,3)** | *1* | *2* | *3* | *2* | 3 | *10* | *10* | |
| **(3,4)** | *2* | *3* | *2* | *3* | *10* | *10* | *10* | |
| | *A* | *G#* | *G* | *F#* | *F* | *E* | *Eb* | |
| **(4,1)** | *10* | *10* | *0* | *1* | 2 | 3 | 2 | 3 |
| **(4,2)** | *10* | *0* | *1* | *2* | 3 | 2 | 3 | *10* |
| **(4,3)** | *0* | *1* | *2* | *3* | 2 | 3 | *10* | *10* |
| **(4,4)** | *1* | *2* | *3* | *2* | 3 | *10* | *10* | *10* |
| | *Bb* | *A* | *G#* | *G* | *F#* | *F* | *E* | *Eb* |

**Figure 8**. Table containing the individual costs of playing a note a certain number of half steps away with a pair of fingers in 3.5 Position. The column on the far left under "Fingers (start, end)" contains the current finger we have pressed down and the next finger that could be pressed, whereas under rows, we have the "# or Number of Half-Steps", which documents how far away the next note is. The data values are the 'costs 'that come with the pair of fingers, ranging from 0 (being easy) to 4 (being hard) to 10 (being not conventionally done).

It can be seen that with each increment in the starting finger, there is an increment in the number of half steps columns because the distance to 0.5 position increases. There are a total of 12 tables for this system, 6 tables (one representing position) for each of the two components of shifting.

## The Dynamic Programming Approach

Dynamic programming is a systematic approach to a problem that breaks down the approach into a variety of sequential steps that are optimized in the process. A popular example is the 'Traveling Salesman Problem 'where a salesman needs to travel to different cities exactly once and return to his starting city after visiting all the cities. In order to find the optimal route, we need to break the process into sequential steps. The first step would be to find the closest city from the salesman's current location and move the salesman's current location to that closest city; after that, we need to find the closest city from the new current city and so on until we've traveled to all the cities. With this process, we find the optimal route at every single step and arrive at the route that is the most efficient. A similar technique is applied to finding the best fingering–the only exception is that the salesman now has 4 different starting points to choose from and 16 different ways to get to a city, in which the difficulty of each way of transportation varies from city to city. Virtually, the salesman becomes our hand and the cities become our fingers.

The approach to finding the optimal fingering begins with the starting finger and the current note. Since most of the time cellists use only 4 fingers up to 4th Position, we will have 4 different starting positions. Utilizing the procedure in finding the nearest city with The Traveling Salesman, we can find the next best finger to place given a starting finger. Below is a general equation encompassing this concept and meaning for each abbreviation:

DP(n, f) = min(DP(n+1, g) + cost(n, f, n+1, g) for g in 1...4)

DP(n, f) - *the entire dynamic programming algorithm that takes in the current note n and the the starting finger f and begins the process in finding the best fingering*

min(DP(n+1, g) - *the function that takes in the next note in the sequence n + 1 and the the next best finger g*

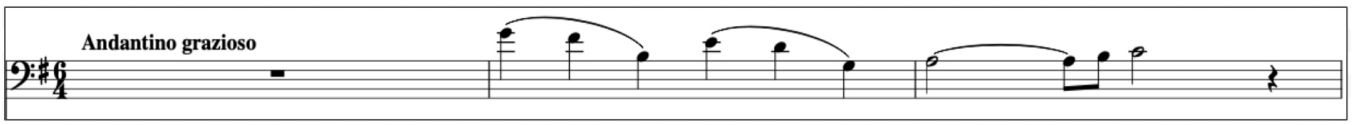
cost(n, f, n+1, g) - *the cost function that returns the cost given the current note n , the starting finger f , the next note n + 1 and the next finger g*

for g in 1...4 - *the programming statement that increments the next finger g from 1 to 4*

With 4 starting fingers we will end up having 4 of these equations and therefore solutions:

DP (n, 1) = min (DP (n+1, g) + cost (n,1, n+1, g) for g in 1...4)
DP (n, 2) = min( DP (n+1, g) + cost (n, 2, n+1, g) for g in 1...4)
DP (n, 3) = min (DP (n+1, g) + cost (n, 3, n+1, g) for g in 1...4)
DP (n, 4) = min (DP (n+1, g) + cost (n, 4, n+1, g) for g in 1...4)

For instance in 'The Swan 'by Camille Saint Saens, we will start the optimization procedure with fingers 1, 2, 3 & 4




(see Figure 9).

**Figure 9.** Example of The Swan 'by Camille Saint Saens, a renowned piece for cellists of all levels and ages.

The algorithm then goes through the 16 possible fingering combinations given the distance to the next note and finds the next best finger to put down, ultimately arriving at a set of fingerings for each starting finger. Below includes the 4 sets of fingerings and 4 costs associated with each fingering. The starting finger is highlighted in yellow.

Best Fingers: [1, 1, 1, 4, 2, 1, 2, 4, 4]
Best Cost: 23
Best Fingers: [2, 1, 1, 4, 2, 1, 2, 4, 4]
Best Cost: 22
Best Fingers: [3, 2, 1, 4, 2, 1, 2, 4, 4]
Best Cost: 15
Best Fingers: [4, 3, 1, 4, 2, 1, 2, 4, 4]
Best Cost: 14

Once we have these 4 sets, we have to find the set that is the least costly, which is the last set, starting with finger 4. The algorithm then returns the optimal fingering for this musical passage:

Your optimal fingering: [4, 3, 1, 4, 2, 1, 2, 4, 4]

| IV | | III | | II | | I | |
|---|---|---|---|---|---|---|---|
| 0 | C2 | 12 | G2 | 24 | D3 | 36 | A3 |
| 1 | C#2 | 13 | G#2 | 25 | D#2 | 37 | A#3 |
| 2 | D2 | 14 | A2 | 26 | E3 | 38 | B3 |
| 3 | D#2 | 15 | A#2 | 27 | F3 | 39 | C4 |
| 4 | E2 | 16 | B2 | 28 | F#3 | 40 | C#4 |
| 5 | F2 | 17 | C3 | 29 | G3 | 41 | D4 |
| 6 | F#2 | 18 | C#3 | 30 | G#2 | 42 | D#4 |
| 7 | G2 | 19 | D3 | 31 | A3 | 43 | E4 |
| 8 | G#2 | 20 | D#3 | 32 | A#3 | 44 | F4 |
| 9 | A2 | 21 | E3 | 33 | B3 | 45 | F#4 |
| 10 | A#2 | 22 | F3 | 34 | C4 | 46 | G4 |
| 11 | B2 | 23 | F#3 | 35 | C#4 | 47 | G#4 |

## Implementation

**Figure 10.** A diagram showing my 'notes numbering system', where each note on the cello (until 4th position) is given a number. This system enables the algorithm to calculate the distance between each note and the position the player is currently in.

To understand the algorithm itself in detail, we would first need to understand the notes numbering system. In Figure 10 there are Roman numerals I, II, III, & IV, representing the cello strings A, D, G, & C respectively.

Given this, we can assign the notes for 'The Swan 'to their corresponding numbers. The reason for this is because we need to determine the current position that we are in in order to choose the correct table of costs. Below is a diagram illustrating a birds eye view of what occurs in the algorithm.

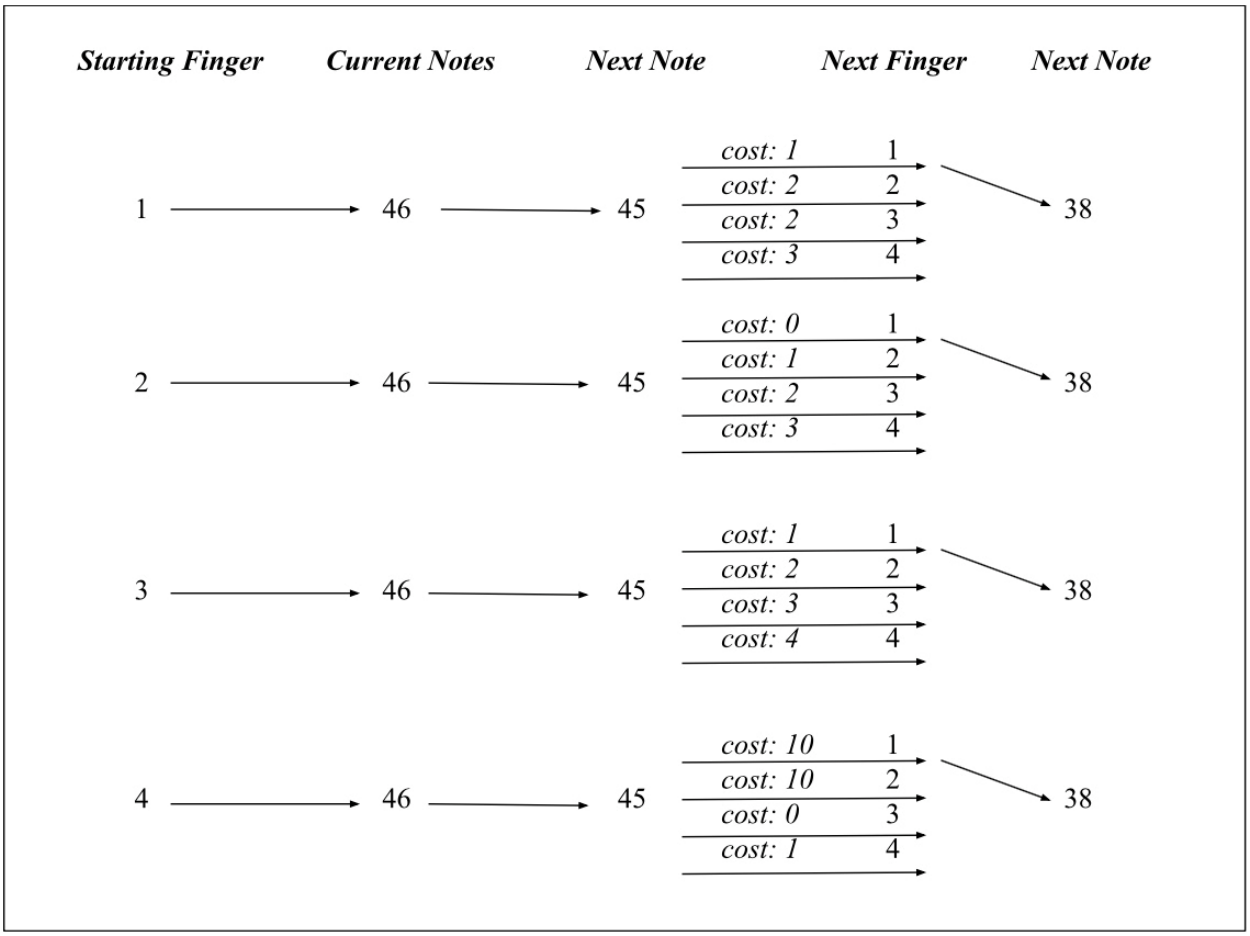



**Figure 11.** A visual representation of the process the algorithm goes through.

With our starting finger, we would first determine the position we are in and the current note to be played. Then, we scan for the next note and iterate through all the possible options. Recall in this set of equations (below), we find the best next finger to put down, 'g', by going into the tables and finding the cost of the shift. We then find the next finger (boxed in the diagram) and move on to the next note (see Figure 11).

DP(n, 1) = min(DP(n+1, g) + cost(n,1, n+1, g) for g in 1...4)
DP(n, 2) = min(DP(n+1, g) + cost(n, 2, n+1, g) for g in 1...4)
DP(n, 3) = min(DP(n+1, g) + cost(n, 3, n+1, g) for g in 1...4)
DP(n, 4) = min(DP(n+1, g) + cost(n, 4, n+1, g) for g in 1...4)

After determining the best possible set of fingerings with each starting finger, we then compare their individual costs and present the user with the best possible fingering.

## Limitations

While this algorithm successfully returns optimal fingerings, it only calculates the immediate next best step and lacks the ability to 'foresee 'the possibility of perhaps using a costly finger to lessen the total cost in the long run. Similarly, this algorithm fails to factor in the possibility of overlapping notes and the benefits it may pose to the entire trajectory of the piece. On the cello there are 15 notes that can be played in two different positions, however the algorithm only chooses the most common and seemingly easiest position: 1st position. As stated, occasionally attempting a harder shift may enable the cellist to avoid excess shifting or stretching. Furthermore, for pieces such as sonatas or concertos that require the higher registers of a cello, thus involving the "thumb position" where cellists use their thumb to play, this algorithm is not applicable, as it is restricted to the 4th position. By delving into the thumb position, a new world of calculations will surface with the dramatic increase in the number of notes. For reference, there are 47 notes that can be played up to 4th position and approximately 128 notes on the entire cello (it depends on the length of the fingerboard which varies from luthier to luthier). The number of positions one single note can be played on will increase, adding layers of complexity to the algorithm. However, in conquering these limitations in the future, this algorithm will not only become more accurate, but will be a tool for cellists of all ages and levels to utilize.

## Conclusion

Although this dynamic programming algorithm has its shortcomings, it not only revolutionizes the process of learning a string instrument but presents us with the possibility to encourage more students to learn music on their own, not needing to wait for a teacher to give them a fingering and invariably preventing beginners from running into obstacles and becoming discouraged. This algorithm will help students, teachers, and performers alike to save time when learning a new piece or double checking their fingering with this generator. Our time spent on testing out fingerings will be saved and turned into more time to enjoy playing the instrument.

## Acknowledgements

Thank you to my cello teacher Jonathan Koh for inspiration and Qi Xuan Khoo for all his guidance.

## References

Authored by - diva. (n.d.). https://www.diva-portal.org/smash/get/diva2:768564/FULLTEXT01.pdf
*A simplified guide to dynamic programming*. (2022, October 19). Spiceworks. https://www.spiceworks.com/tech/devops/articles/what-is-dynamic-programming/

YouTube. (2013, January 14). *Lecture 22: Dynamic Programming IV: Guitar Fingering, Tetris, super Mario Bros.*.YouTube. https://www.youtube.com/watch?v=tp4_UXaVyx8

Division of Geological and Planetary Sciences. (n.d.-b). https://www.gps.caltech.edu/~tsai/files/HartBoschTsai_2000.pdf

*Selected notable sales - darnton & hersh fine violins: Chicago violin sales and Restoration*. Darnton & Hersh Fine Violins | Chicago Violin Sales and Restoration. (2022, August 29). https://darntonhersh.com/notable-sales/

Parncutt, R., Sloboda, J., Clarke, E., Raekallio, M. and Desain, P. (1997)"An ergonomic model of keyboard fingering for melodic fragments," *Music Perception*, Vol.14, pp. 341 - 382

Yonebayashi, Y., Kameoka, H. and Sagayama,S.(2010)"Automatic Decision of Piano Fingering Based on Hidden Markov Models"