Author: Tim Baron
12/14/2018

Optimization of a Guitar Fretboard

Short Abstract:

This project optimizes movement between major triad chords in the key of A between the 2nd and 12th fret and optimizes the distance between notes for a song.

Six optimized chords were determined with a total distance of 60.23 inches. The minimal path for the song was 4.083 inches.

Medium Abstract:

Studying notes and chords on the guitar fretboard is a complicated task. There is much debate about the most effective and/or efficient way to play and learn the fretboard.

Mapping the fretboard and creating exercises is a highly effective way to learn to use the fretboard. Two common learning strategies are the CAGED method and the Planetalk method. These methods view the fretboard as a series of numbers that represent the notes of the major scale. When looking at the fretboard map one can see the individual numbers of the scale notes and the chord possibilities.

The goal of this project was to optimize, in this case minimize, the movement between all of the major triad chords in the key of A between the 2nd and 12th fret using a traveling salesperson (TSP) model and to minimize the distance between notes to play a simple song using a network flow model.

Both methods were successful in determining minimal paths. Six optimized chords were determined with a minimized distance of 60.23 inches. A distance of 4.083 inches was determined as the minimal distance path between notes when playing the simple song.

Introduction:

This project will analyze the guitar fretboard using mathematical optimization methods. Distances between fretted notes on a guitar will be measured and recorded for the chord

model. Distances between notes for the simple song model will be calculating using a formula and will ignore the fact that frets get smaller as one moves up the neck. These distances will be the raw data used for optimization.

The main goals of this project are:

- Determine six optimized three note major triad chords from the 2nd fret to the 12th fret in the key of A.
- Compare the distance cost of the optimized chords to some known standard chords.
- Optimize the playing of a simple song.
- Debug/test the spreadsheet formulas/constraints.

Other considerations:

- Note bending and barre chords are being ignored in this project. They would be suitable for a more complex and in-depth project.
- We will not be concerned with different octave notes being combined. Combined octave notes are acceptable because they are part of the "art" of using chords.
- Two notes cannot be played at the same time on the same string, thus we must create constraints to prevent this, if necessary.
- We will assume that the guitar player is starting from the headstock (the left side of the diagram) and playing up the neck (to the right on the diagram) for optimization purposes.

Methods, Models, and Calculations:

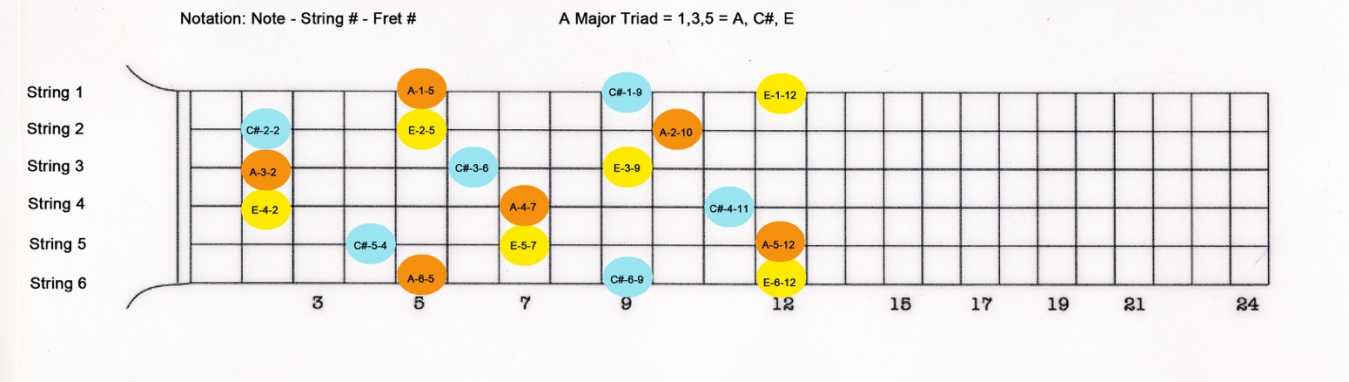
Excel Solver will be used for the analysis.

Method/Model – Optimized Major Triads – TSP:

The optimization of major triad chords between the 2nd and 12th fret was analyzed using a Traveling Sales Person (TSP) model. Since there are three notes in a triad, there were three TSPs, one for each note, and the results are stacked to form the optimized chords. This stacked solution will be compared to a stacked solution containing common/expected A major triad chords between the 2nd and 12th fret found in a standard guitar chord encyclopedia.

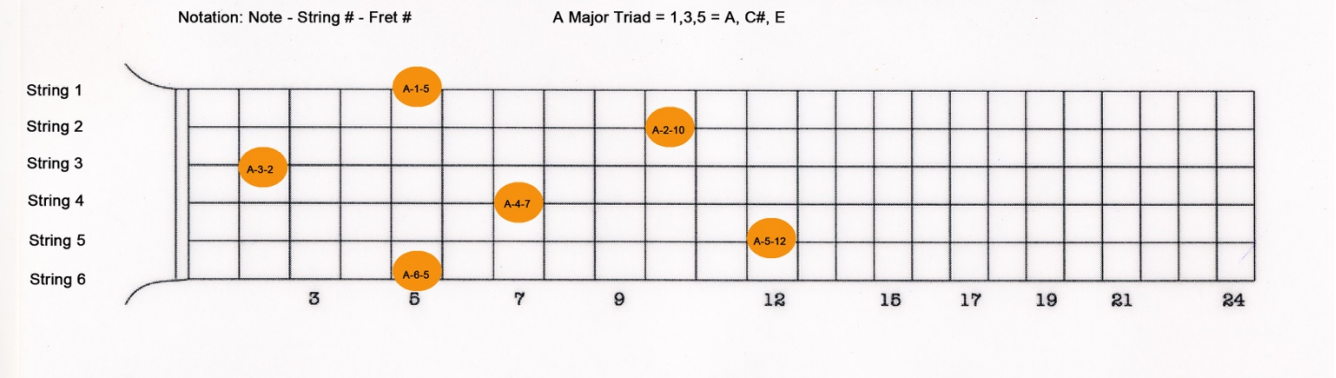
Chord/Fretboard Diagrams:

The notes for the chords to be studied are outlined in the following diagram in A major. The notation for each note is read as note-string-fret. For example, A-3-2 is the note A on the third string, second fret.



The distance between each note was measured using a standard tape measure accurate to 1/16th of an inch.

For the major triad chord TSP, three data tables were created. Each table is for a specific note (A, C#, or E). For example, the “A” table below, measures the distance between each “A” on the fretboard.



*measurements are in inches

	A-3-2	A-1-5	A-6-5	A-4-7	A-2-10	A-5-12
A-3-2	0.00	3.50	3.63	5.63	8.25	9.88
A-1-5	3.50	0.00	1.56	2.38	4.69	6.38
A-6-5	3.63	1.56	0.00	2.25	4.94	6.31
A-4-7	5.63	2.38	2.25	0.00	2.75	4.25
A-2-10	8.25	4.69	4.94	2.75	0.00	1.88
A-5-12	9.88	6.38	6.31	4.25	1.88	0.00

This table is then combined with a decision variable matrix, shown below, using the =sumproduct Excel function to form a TSP calculation in Solver.

	A-3-2	A-1-5	A-6-5	A-4-7	A-2-10	A-5-12	Sum
A-3-2	0	1	0	0	0	0	1
A-1-5	0	0	0	0	1	0	1
A-6-5	1	0	0	0	0	0	1
A-4-7	0	0	1	0	0	0	1
A-2-10	0	0	0	0	0	1	1
A-5-12	0	0	0	1	0	0	1
Sum	1	1	1	1	1	1	

The rows and columns are summed so that they can be used as constraints in the TSP calculation. Each row and column must add up to “1” to ensure that a note is only used one time. The decision variables are constrained to be binary. The diagonal of zeroes prevents a note from traveling to itself. The Excel Solver method used is Simplex LP.

There are three data tables, three decision variable matrices, and three objective functions – one for each note. To optimize all of the sub-objective functions at once, the objective function used in solver is the sum of the three objective functions. The setup is as shown below:

A

*measurements are in inches

	A-3-2	A-1-5	A-6-5	A-4-7	A-2-10	A-5-12
A-3-2	0.00	3.50	3.63	5.63	8.25	9.88
A-1-5	3.50	0.00	1.56	2.38	4.69	6.38
A-6-5	3.63	1.56	0.00	2.25	4.94	6.31
A-4-7	5.63	2.38	2.25	0.00	2.75	4.25
A-2-10	8.25	4.69	4.94	2.75	0.00	1.88
A-5-12	9.88	6.38	6.31	4.25	1.88	0.00

Obj. Funct:	20.2
Recorded:	21.82

	A-3-2	A-1-5	A-6-5	A-4-7	A-2-10	A-5-12	Sum
A-3-2	0	1	0	0	0	0	1
A-1-5	0	0	0	0	1	0	1
A-6-5	1	0	0	0	0	0	1
A-4-7	0	0	1	0	0	0	1
A-2-10	0	0	0	0	0	1	1
A-5-12	0	0	0	1	0	0	1
Sum	1	1	1	1	1	1	

Constraints:
Decision variables binary
location to same location = 0
row sum = 1
column sum = 1

Tour:	A-3-2	A-1-5	A-2-10	A-5-12	A-4-7	A-6-5	A-3-2

Subtour Constraints:	LHS		RHS
A-3-2 , A-1-5	1	<=	1
A-6-5 , A-4-7	1	<=	1
A-2-10 , A-5-12	1	<=	1
A-3-2 , A-1-5, A-5-12	1	<=	2
A-3-2, A-1-5, A-6-5, , A-5-12	2	<=	3
A-3-2, A-1-5, A-6-5	2	<=	2

C#

*measurements are in inches

	C#-2-2	C#-5-4	C#-3-6	C#-1-9	C#-6-9	C#-4-11
C#-2-2	0.00	2.69	4.63	7.38	7.50	9.00
C#-5-4	2.69	0.00	2.31	5.13	5.00	6.63
C#-3-6	4.63	2.31	0.00	2.88	3.00	4.44
C#-1-9	7.38	5.13	2.88	0.00	1.69	1.88
C#-6-9	7.50	5.00	3.00	1.69	0.00	1.75
C#-4-11	9.00	6.63	4.44	1.88	1.75	0.00

Obj. Funct:	18.83
Recorded:	19.01

	C#-2-2	C#-5-4	C#-3-6	C#-1-9	C#-6-9	C#-4-11	Sum
C#-2-2	0	1	0	0	0	0	1
C#-5-4	0	0	0	0	1	0	1
C#-3-6	1	0	0	0	0	0	1
C#-1-9	0	0	1	0	0	0	1
C#-6-9	0	0	0	0	0	1	1
C#-4-11	0	0	0	1	0	0	1
Sum	1	1	1	1	1	1	

Constraints:
Decision variables binary
location to same location = 0
row sum = 1
column sum = 1

Tour:	C#-2-2	C#-5-4	C#-6-9	C#-4-11	C#-1-9	C#-3-6	C#-2-2

Subtour Constraints:	LHS		RHS
A-3-2 , A-1-5	1	<=	1
A-6-5 , A-4-7	1	<=	1
A-2-10 , A-5-12	1	<=	1
A-3-2 , A-1-5, A-5-12	1	<=	2
C#-2-2, C#-3-6, C#-5-4	2	<=	2

E

*measurements are in inches

	E-4-2	E-2-5	E-5-7	E-3-9	E-1-12	E-6-12
E-4-2	0.00	3.56	5.63	7.38	9.75	9.81
E-2-5	3.56	0.00	2.25	3.81	6.31	6.38
E-5-7	5.63	2.25	0.00	1.94	4.38	4.19
E-3-9	7.38	3.81	1.94	0.00	2.38	2.63
E-1-12	9.75	6.31	4.38	2.38	0.00	1.63
E-6-12	9.81	6.38	4.19	2.63	1.63	0.00

Obj. Funct:	21.2
Recorded:	21.2

	E-4-2	E-2-5	E-5-7	E-3-9	E-1-12	E-6-12	Sum
E-4-2	0	1	0	0	0	0	1
E-2-5	0	0	0	1	0	0	1
E-5-7	1	0	0	0	0	0	1
E-3-9	0	0	0	0	1	0	1
E-1-12	0	0	0	0	0	1	1
E-6-12	0	0	1	0	0	0	1
Sum	1	1	1	1	1	1	

Constraints:
Decision variables binary
location to same location = 0
row sum = 1
column sum = 1

Tour:	E-4-2	E-2-5	E-3-9	E-1-12	E-6-12	E-5-7	E-4-2

Subtour Constraints:	LHS		RHS
A-3-2 , A-1-5	1	<=	1
A-6-5 , A-4-7	0	<=	1
A-2-10 , A-5-12	1	<=	1
A-3-2 , A-1-5, A-5-12	1	<=	2
E-4-2, E-5-7, E-2-5	2	<=	2

Above are all three data tables, their individual constraints, and the resulting TSP tours. The global objective function (shown below) is the sum of the three objective functions shown above.

Global Obj. Func:	60.23

Global Info:

Decision Vars:	b37:g42,b67:g72,b97:g102

Global Constraints (to prevent two notes on one string):

Notes	LHS		RHS
C#-5-4, E-5-7	2	<=	1
C#-6-9, E-6-12	2	<=	1
A-6-5, C#-6-9	2	<=	1
A-5-12, E-5-7	2	<=	2
A-6-5, E-6-12	2	<=	2
C#-1-9, E-1-12	2	<=	2
C#-3-6, E-3-9	2	<=	2
A-5-12, C#-5-4	2	<=	2
A-6-5, C#-6-9, E-6-12	3	<=	3
C#-1-9, E-1-12	2	<=	2
C#-3-6, E-3-9	2	<=	2
A-5-12, C#-5-4	2	<=	2
String 1	3	<=	3
String 2	3	<=	3
String 3	3	<=	3
String 4	3	<=	3
String 5	3	<=	3
String 6	3	<=	3

There are both individual TSP constraints and global constraints. The individual constraints establish that the decision variables are binary, the same-to-same location is always equal to zero, and that the row and columns always sum to one. In addition, subtours showed up when calculating the TSP. In these cases, a note would travel to another note and then end up traveling back to itself, thus duplicating something that was already used. Since each chord needs to be unique this was not acceptable. Constraints for eliminating the subtours were determined by analyzing the data after running solver. As subtours appeared, constraints were created to prevent the subtours. This process was repeated until all subtours were eliminated.

Global constraints were used to prevent two different notes (for example, A and E) from being on the same string when played in a chord - two notes being played on the same string at the same time is not possible. Solver was run multiple times to determine if any same-note-string assignments showed up. If same-note-string assignments were present, additional constraints were added to eliminate them.

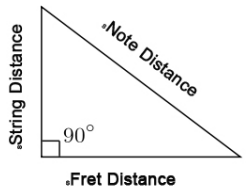
Method/Model – Playing a Simple Song – Network Flow:

Optimization to minimize the distance between the notes when playing a simple song was done using a network flow model. For this model, distance was determined using a distance formula:

Distance between strings = 0.313 inches

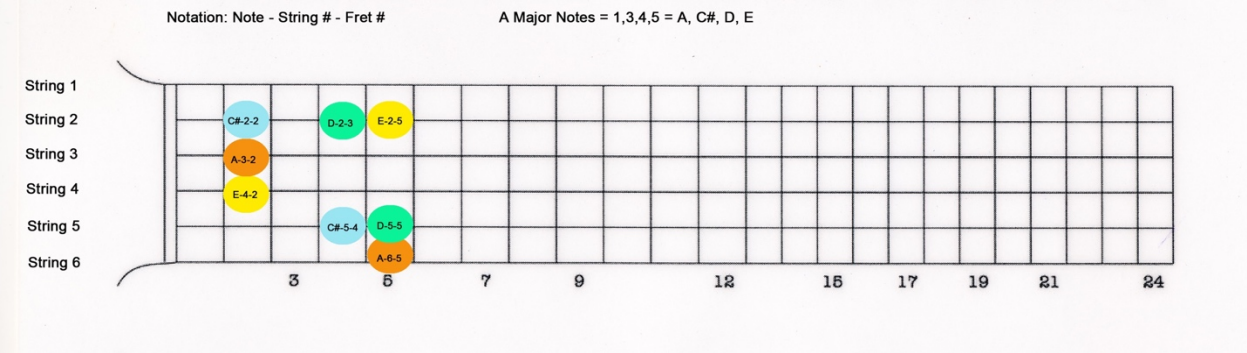
Distance between frets = 1.75 inches

F = # frets
S = # strings
D = note distance



$$D = \sqrt{(1.75F)^2 + (0.313S)^2}$$

The notes of the song to be optimized are (in order): A D A C# E A (without regard to octave). The distance between each possible note change was determined using the formula above and used as "distance cost" in the flow model. For example, A32 – D23, is the note change between the "A" note on the 3rd string 2nd fret and the "D" note on the 2nd string 3rd fret and the distance for that change is 3.51 inches. The diagrams below show the available notes, the decision variables, and the distance costs.



Paths	A32 - D23	A32 - D55	A65 - D23	A65 - D55	A32 - D23	A32 - D55	A65 - D23	A65 - D55	A32 - C#22	A32 - C#54	A65 - C#22	A65 - C#54	C#22 - E42	C#22 - E25	C#54 - E42	C#54 - E25	A32 - E42	A32 - E25	A65 - E42	A65 - E25
Dec. Vars	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1
Distance Cost	3.51	5.29	2.15	0.313	3.51	5.29	2.15	0.313	0.313	3.56	5.4	1.78	0.626	5.25	3.51	1.99	0.313	5.26	5.29	1.25

A challenge with this model was that notes in the path had to be linked to each other. For example, when going from C# to E, to get from A65 – C#54 (which would be arriving to C#54 from A65) to the next note in the song, "E", one can only land on a note that has a C#54 and an "E". Thus, A65 – C#54 could land on C#54 – E42 or C#54 – E25, but not on C#22 – E42 or C#22 – E25 because those lack the C#54.

This challenge was overcome by using a node-node matrix and numerous constraints within Solver. The matrix is shown below.

												Cosntraints:		
Dec Vars	A-3-2	A-6-5	D-2-3	D-5-5	A-3-2	A-6-5	C#-2-2	C#-5-4	E-4-2	E-2-5	Sum	LHS		RHS
A-3-2	0	0	0	0	0	0	0	0	0	0	0	1	"="	1
A-6-5	0	0	0	1	0	0	0	0	0	0	1			
D-2-3	0	0	0	0	0	0	0	0	0	0	0	1	"="	1
D-5-5	0	1	0	0	0	0	0	0	0	0	1			
A-3-2	0	0	0	0	0	0	0	0	0	0	0	1	"="	1
A-6-5	0	0	0	0	0	0	0	1	0	0	1			
C#-2-2	0	0	0	0	0	0	0	0	0	0	0	1	"="	1
C#-5-4	0	0	0	0	0	0	0	0	0	1	1			
E-4-2	0	0	0	0	0	0	0	0	0	0	0	1	"="	1
E-2-5	0	0	0	0	0	1	0	0	0	0	1			
Sum	0	1	0	1	0	1	0	1	0	1				

	A-3-2	A-6-5	D-2-3	D-5-5	A-3-2	A-6-5	C#-2-2	C#-5-4	E-4-2	E-2-5
A-3-2	0	0	3.51	5.29	0	0	0	0	0	0
A-6-5	0	0	2.15	0.313	0	0	0	0	0	0
D-2-3	0	0	0	0	3.51	2.15	0	0	0	0
D-5-5	0	0	0	0	5.29	0.313	0	0	0	0
A-3-2	0	0	0	0	0	0	0.313	3.56	0	0
A-6-5	0	0	0	0	0	0	5.4	1.78	0	0
C#-2-2	0	0	0	0	0	0	0	0	0.626	5.25
C#-5-4	0	0	0	0	0	0	0	0	3.51	1.99
E-4-2	0.313	5.29	0	0	0	0	0	0	0	0
E-2-5	5.26	1.25	0	0	0	0	0	0	0	0

The constraints were created by confining each row with the same note to only one occurrence with sum formulas (in yellow above). Notes that were not part of the song note path were constrained to zero for each row (for example, in rows one and two, the only available next note is D). The row sums and column sums ensure that each note is played once.

Results:

Stacked Chord – TSP Result:

Stacked Solution:

	A Chord 1	A Chord 2	A Chord 3	A Chord 4	A Chord 5	A Chord 6
A	A-3-2	A-1-5	A-2-10	A-5-12	A-4-7	A-6-5
C#	C#-2-2	C#-5-4	C#-6-9	C#-4-11	C#-1-9	C#-3-6
E	E-4-2	E-2-5	E-3-9	E-1-12	E-6-12	E-5-7

Very Common Chord Fingering

Two Notes on Same String

Reasonable, but uncommon

Less reasonable, but doable

Completely unreasonable

Global Obj. Func:	60.23

The final stacked solution for the three TSPs showed six playable/feasible chords as shown above. The summed objective function was compared to the summed objective function for some standard chords that would be expected to show up for an A major triad between the 2nd and 12th fret. The optimized solution cost less in distance than did the forced standard chord solution as can be seen below.

Stacked Solution (standard):

	A Chord 1	A Chord 2	A Chord 3	A Chord 4	A Chord 5	A Chord 6
A	A-1-5	A-6-5	A-3-2	A-4-7	A-5-12	A-2-10
C#	C#-5-4	C#3-6	C#-2-2	C#-6-9	C#-4-11	C#-1-9
E	E-2-5	E-5-7	E-4-2	E-1-12	E-6-12	E-3-9

Very Common Chord Fingering

Two Notes on Same String

Reasonable, but uncommon

Less reasonable, but doable

Completely unreasonable

Recorded Optimum:	60.23
Global Obj. Func:	70.54

The standard chords used in the diagram above can be found in the *Mel Bay Deluxe Guitar Chord Encyclopedia* (see bibliography) on page 110. In addition, the standard chords "A Chord 3" and "A Chord 6" are part of both *Fretboard Logic* (CAGED system) and the *Planetalk System* (see bibliography for both).

Playing a Simple Song – Network Flow Result:

The final result for playing a simple song consisting of the notes "A D A C# E A" from our fretboard was A65 to D55 to A65 to C#54 to E25 to A65 with a total distance of 4.083 inches.

Debugging:

Both sheets were copied and debugged. In both cases, data was manually input into the decision variable cells to force an obvious result. Initially some bugs showed up in the network flow model and it was tweaked until they were fixed.

Alternate distances, ones in which the optimized answer would be obvious, were entered into the matrices for each model to be sure that they gave the obvious answer.

Original Distances TSP:

	A-3-2	A-1-5	A-6-5	A-4-7	A-2-10	A-5-12
A-3-2	0.00	3.50	3.63	5.63	8.25	9.88
A-1-5	3.50	0.00	1.56	2.38	4.69	6.38
A-6-5	3.63	1.56	0.00	2.25	4.94	6.31
A-4-7	5.63	2.38	2.25	0.00	2.75	4.25
A-2-10	8.25	4.69	4.94	2.75	0.00	1.88
A-5-12	9.88	6.38	6.31	4.25	1.88	0.00

Obj. Funct:	21.64
Recorded:	20.2

	A-3-2	A-1-5	A-6-5	A-4-7	A-2-10	A-5-12	Sum
A-3-2	0	0	0	1	0	0	1
A-1-5	0	0	1	0	0	0	1
A-6-5	1	0	0	0	0	0	1
A-4-7	0	0	0	0	0	1	1
A-2-10	0	1	0	0	0	0	1
A-5-12	0	0	0	0	1	0	1
Sum	1	1	1	1	1	1	

Tour:	A-1-5	A-6-5	A-3-2	A-4-7	A-5-12	A-2-10	A-1-5

Alternate Distances TSP:

	A-3-2	A-1-5	A-6-5	A-4-7	A-2-10	A-5-12			Obj. Funct:	20.2
A-3-2	0.00	3.50	5.00	6.00	7.00	8.00				
A-1-5	5.00	0.00	6.00	7.00	4.69	8.00			Recorded:	21.82
A-6-5	3.63	8.00	0.00	7.00	6.00	5.00				
A-4-7	8.00	7.00	2.25	0.00	6.00	5.00				
A-2-10	4.00	5.00	6.00	7.00	0.00	1.88				
A-5-12	8.00	7.00	6.00	4.25	5.00	0.00				

	A-3-2	A-1-5	A-6-5	A-4-7	A-2-10	A-5-12	Sum
A-3-2	0	1	0	0	0	0	1
A-1-5	0	0	0	0	1	0	1
A-6-5	1	0	0	0	0	0	1
A-4-7	0	0	1	0	0	0	1
A-2-10	0	0	0	0	0	1	1
A-5-12	0	0	0	1	0	0	1
Sum	1	1	1	1	1	1	

Tour:	A-3-2	A-1-5	A-2-10	A-5-12	A-4-7	A-6-5	A-3-2

Original Distances Node-to-Node:

Dec Vars	A-3-2	A-6-5	D-2-3	D-5-5	A-3-2	A-6-5	C#-2-2	C#-5-4	E-4-2	E-2-5
A-3-2	0	0	0	0	0	0	0	0	0	0
A-6-5	0	0	0	1	0	0	0	0	0	0
D-2-3	0	0	0	0	0	0	0	0	0	0
D-5-5	0	1	0	0	0	0	0	0	0	0
A-3-2	0	0	0	0	0	0	0	0	0	0
A-6-5	0	0	0	0	0	0	0	1	0	0
C#-2-2	0	0	0	0	0	0	0	0	0	0
C#-5-4	0	0	0	0	0	0	0	0	0	1
E-4-2	0	0	0	0	0	0	0	0	0	0
E-2-5	0	0	0	0	0	1	0	0	0	0
Sum	0	1	0	1	0	1	0	1	0	1

	A-3-2	A-6-5	D-2-3	D-5-5	A-3-2	A-6-5	C#-2-2	C#-5-4	E-4-2	E-2-5
A-3-2	0	0	3.51	5.29	0	0	0	0	0	0
A-6-5	0	0	2.15	0.313	0	0	0	0	0	0
D-2-3	0	0	0	0	3.51	2.15	0	0	0	0
D-5-5	0	0	0	0	5.29	0.313	0	0	0	0
A-3-2	0	0	0	0	0	0	0.313	3.56	0	0
A-6-5	0	0	0	0	0	0	5.4	1.78	0	0
C#-2-2	0	0	0	0	0	0	0	0	0.626	5.25
C#-5-4	0	0	0	0	0	0	0	0	3.51	1.99
E-4-2	0.313	5.29	0	0	0	0	0	0	0	0
E-2-5	5.26	1.25	0	0	0	0	0	0	0	0

Solution:

A65 to D55 to A65 to C#54 to E25 to A65

Alternate Distances Node-to-Node:

Dec Vars	A-3-2	A-6-5	D-2-3	D-5-5	A-3-2	A-6-5	C#-2-2	C#-5-4	E-4-2	E-2-5
A-3-2	0	0	0	0	0	0	0	0	0	0
A-6-5	0	0	1	0	0	0	0	0	0	0
D-2-3	0	1	0	0	0	0	0	0	0	0
D-5-5	0	0	0	0	0	0	0	0	0	0
A-3-2	0	0	0	0	0	0	0	0	0	0
A-6-5	0	0	0	0	0	0	0	1	0	0
C#-2-2	0	0	0	0	0	0	0	0	0	0
C#-5-4	0	0	0	0	0	0	0	0	1	0
E-4-2	0	0	0	0	0	1	0	0	0	0
E-2-5	0	0	0	0	0	0	0	0	0	0
Sum	0	1	1	0	0	1	0	1	1	0

	A-3-2	A-6-5	D-2-3	D-5-5	A-3-2	A-6-5	C#-2-2	C#-5-4	E-4-2	E-2-5
A-3-2	0	0	8	8	0	0	0	0	0	0
A-6-5	0	0	1	8	0	0	0	0	0	0
D-2-3	0	0	0	0	8	1	0	0	0	0
D-5-5	0	0	0	0	8	8	0	0	0	0
A-3-2	0	0	0	0	0	0	8	8	0	0
A-6-5	0	0	0	0	0	0	8	1	0	0
C#-2-2	0	0	0	0	0	0	0	0	8	8
C#-5-4	0	0	0	0	0	0	0	0	1	8
E-4-2	1	8	0	0	0	0	0	0	0	0
E-2-5	8	8	0	0	0	0	0	0	0	0

Solution:
A65 to D23 to A65 to C#54 to E42 to A65

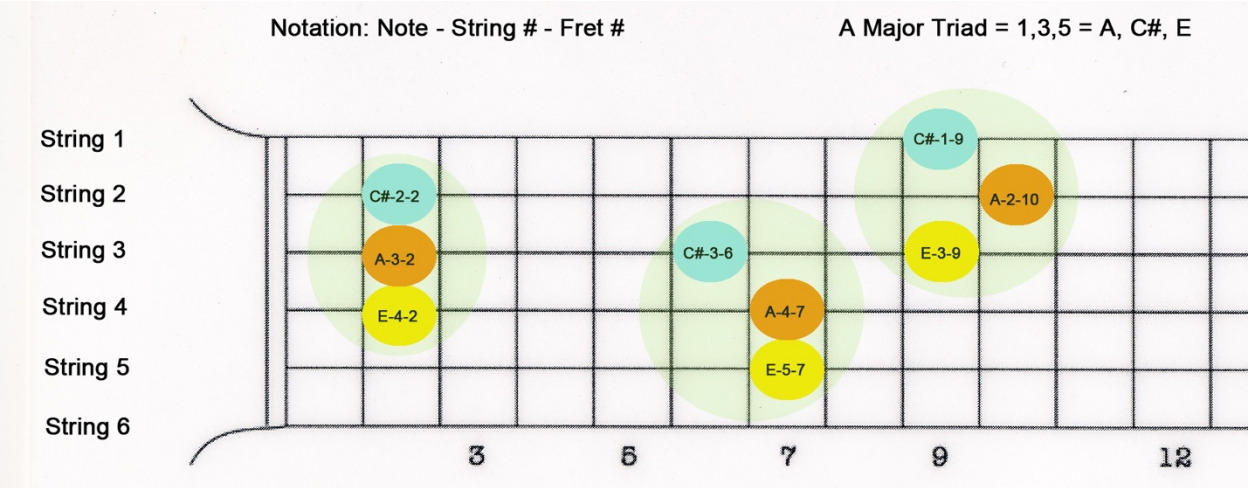
In both cases, the sheets ultimately passed the forced result(s) debug test.

Conclusion:

In conclusion, both models produced feasible results.

I compared my results to a method used in a paper titled *Path Difference Learning for Guitar Fingering Problem* (see bibliography). In the paper they do a combination of what I've done (chords and single notes) using a path difference learning (PD) and a weighted cost for changes in fret position/movement of strings that they determined from published tablature (a number based system for reading guitar music). I think that in terms of looking at this issue mathematically, they may have obtained superior results.

However, I am going to add something to my conclusion regarding theory versus practice. I was generally interested in the conclusion that would be reached when optimizing the fretboard, but both my mathematical conclusion and the mathematical conclusion from the paper above ignore two things. First, they ignore that the player has to be able to tell where they are on the fretboard to know where they are going (in other words they must be able to travel the fretboard not just physically, but also mentally). Therefore, "optimized" chords may be those that help one see the fretboard better as opposed to how close together they are. One way to do this would be to look at chords that always have the same sequential pattern of 1,3, and 5 vertically (and are on adjacent strings) when looking down at the neck. For example, there are three chord patterns that are always arranged when looking down from the 6th string to the 1st string in which the 5th note will be on top, the 1st note will be in the middle, and the 3rd note will be on the bottom. This is better illustrated with a picture:



Second, they completely ignore improvisation. Improvisation, which is the staple of jazz, blues, country, and rock relies on playing in the moment – in which case knowing where you "are" and where you "could go" are what matter and you have no set path. I only include this because I think it's important to always think about theory and practice and almost every guitarist improvises at some point. As a second side note, having a bunch of optimized paths to use for improvisation could be taken from the mathematical results and be very useful.

Annotated Bibliography:

Path Difference Learning for Guitar Fingering Problem, Aleksander Radisavljevic and Peter Driessen, Department of Electrical and Computer Engineering, University of Victoria, British Columbia

Annotation: This paper was a published study on guitar fretboard optimization. The authors used a path learning model with weighted costs for movement between frets/strings. They had a large sample of songs in their study. They used tablature to determine the chord shapes. However, their study differed from mine in that I was more interested in optimizing the fretboard "in general" (specifically for applied usage) as opposed to optimizing songs already written and played a certain way. However, they're methodology was interesting and could also apply to general usage.

Orman, A. J. and Williams, H. Paul (2004) A survey of different integer programming formulations of the travelling salesman problem. Operational Research working papers, LSEOR 04.67. Department of Operational Research, London School of Economics and Political Science, London, UK.

Annotation: This paper describes several different ways of doing TSP problems. This was important for my project because I was new to formulating TSPs and also had TSPs within TSPs and had more constraints to consider than we had in our homework problem. In addition, I had to solve the subtour problem, which we did not have to do in our homework. The paper gave me more insight into the theory behind TSP's than the standard text did. They concluded that a unifying framework had been established for eight different TSPs. They provided a full example of their work. I found the paper to be thorough and having a numerical example was helpful.

A Genetic Algorithm for the Automatic Generation of Playable Guitar Tablature.
D.R. Tuohy and W.D. Potter - Artificial Intelligence Center University of Georgia Athens, Georgia USA

Annotation: This paper describes a method created by the authors to map tablature on the guitar fretboard. They use a Genetic Algorithm (GA). This paper was important for my project because it gave a nice overview of the successful use of an algorithm that has been applied to playing music. They had good diagrams, a clear explanation of their method, and they used popular songs as examples. It's interesting to note that they came up with viable, yet not necessarily desirable, tablatures for some music – which ties in with my comments about human playability. They conclude that their GA can produce good tablature for any piece of guitar music and that it generally outperforms commercial software for this purpose. A weakness of the work, which I didn't explore, is that they did not take into consideration any note bending. They also did not explore improvisation.

Fretboard Logic SE, Bill Edwards, 1989

Annotation: This book describes the CAGED guitar method. Ths method uses the open chord fingerings (chord shapes) of the movable chords C, A, G, E, D to play a single chord in different positions all the way up the neck. It's been in print since 1983 and is widely used. It doesn't require the ability to read music. It would be interesting to see how this method compares to any mathematical optimization method for moving between chords on the guitar. Two of the standard chord shapes used in my model are used in this method (there are five chords shapes in this method). One of the chords appeared in my optimized model.

Planetalk – The Truly Totally Different Guitar Instruction Booklet, Kirk Lorange, 2009

Annotation: This book offers a method similar to the CAGED method that specifically does not involve reading music, but seeing the notes as numbers in movable scales/chords on the neck. This method approaches learning the fretboard from a "visualization of the entire fretboard" perspective and is very practical in nature. My diagrams were inspired by those used in this book. Two of the standard chord shapes used in my model are used in this method (there are only three chords shapes in this method). One of the chords appeared in my optimized model.

Mel Bay's Deluxe Chord Encyclopedia, William Bay, 1971

Annotation: This is probably the most common guitar chord encyclopedia found on the book shelves of guitarists. This was used to verify the standard chord shape comparison optimization

to my optimization model. It verifies that the standard chords I used are known/common chords.

Introduction to Operations Research, Hillier-Lieberman, 10th Edition,

Annotation: An introductory level operations research book. I used this to look at examples of network flow and traveling salesperson problems. There are good examples in here and it's straightforward.

Operations Research – Applications and Algorithms, Wayne L. Winston, 2004

Annotation: This is a highly regarded operations research book. It's very extensive and goes into more depth than the Hillier-Lieberman book. I used this to look at examples of network flow and traveling salesperson problems.

https://www.premierguitar.com/articles/19390-7-the-guitarists-guide-to-the-caged-system

Annotation: A short article on the caged method. The diagrams shed some light on what the chord forms look like. It's a quick hands-on tutorial.

https://en.wikibooks.org/wiki/Tablature

Annotation: A very brief but clear explanation of tablature. This would be helpful for somebody who has no idea what tablature is.