

# AUTOMATIC FINGERING SYSTEM (AFS)

**Alia Al Kasimi**  
Indiana University  
Computer Science Department  
Lindley Hall 215  
Bloomington, IN 47405-7104  
aalkasim@cs.indiana.edu

**Eric Nichols**  
Indiana University  
Computer Science Department  
Lindley Hall 215  
Bloomington, IN 47405-7104  
epnichol@cs.indiana.edu

**Dr. Christopher Raphael**  
Indiana University  
Informatics Department  
Eigenmann 935  
Bloomington, IN 47405-7104  
craphael@indiana.edu

## ABSTRACT

One of the most important aspects in playing the piano is using the appropriate fingers to facilitate movement and transitions. The fingering arrangement depends to a certain extent on the size of the musician's hand. We have developed an automatic fingering system that, given a sequence of pitches, suggests which fingers should be used. The output can be personalized to agree with the limitations of the user's hand. We also consider this system to be the base of a more complex future system: a score reduction system that will reduce orchestra scores to piano scores. This paper describes:

- "Vertical cost" model: the stretch induced by a given hand position.
- "Horizontal cost" model: transition between two hand positions.
- A system that computes low-cost fingering for a given piece of music.
- A machine learning technique used to learn the appropriate parameters in the models.

**Keywords:** fingering, piano, trellis graph, personalization, horizontal cost, vertical cost.

## 1 PROBLEM DOMAIN

Our goal is to solve basic but fundamental problems in piano fingering and then to expand our system to handle more complex examples. The work in progress takes into account the following musical parameters that affect fingering:

- Pitches performed by the right hand
- Single notes and chords
- Tempo
- Articulation: staccato versus legato
- Pedal

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page.

© 2005 Queen Mary, University of London

## 2 AUTOMATIC FINGERING SYSTEM APPLICATION

The AFS takes a quantized MIDI file as input and outputs a text file specifying a possible fingering for each note in the piece. The system selects the optimum fingering by minimizing the costs associated with chord transitions and hand stretches. The minimization is implemented via a dynamic programming approach based on representing the possible fingerings of the piece as a trellis graph. A trellis graph is a multilayer directed acyclic graph where nodes are only connected between adjacent layers.

We selected Java 1.4 as the main programming language for the fingering system. Musical input is in the format of quantized MIDI files.

### 4.1 The Costs

#### 4.1.1 Vertical Cost

The vertical cost corresponds to the stretch induced by a given hand position, where the value of the cost is proportional to the difficulty of the stretch. The system defines every vertical position in time – when a new pitch is added or deleted – as the combination of pitches that are "live" at that instant.



**Figure 1.** Excerpt from Aufschwung (*Phantasiestuck* op.12) by Schumann



**Figure 2.** Vertical analysis of the same excerpt

The example above (Figure 1) demonstrates this "vertical" concept. The first note, F, is still live when G, C and D are played, as one can see in Figure 2. Likewise, E is still live when G is played. The system will determine the vertical cost depending on the position of the hand at each time slice. In this case the chords involved are: F – F G – F C – F D – E – E G.

The vertical cost of a chord is the sum of the costs of the stretches between adjacent pairs of fingers in the chord. There can be between 0 and 4 pairs of fingers in a chord (from a single note to a chord where all the fingers are involved). The cost of a pair of fingers playing two notes is determined by 1) which two fingers are involved and 2) the distance between the two notes being played by these fingers. That is, for a fingered chord  $fc$ , cost  $C$  over pairs  $P_1, P_2, P_3, P_4$  is given by:

$$C(fc) = \sum_{i=1}^4 C(P_i)$$

where any of the individual terms may be 0 if not applicable.

#### 4.1.2 Horizontal Cost

While the vertical cost only considers a single chord at a time, the horizontal cost function accounts for transitions between two successive chords. For example, in Figure 3 the transition between fingers 2,4 and 3,5 is effortless because moving from finger 2 on E to finger 3 on F is easy, as is the transition between fingers 4 and 5 (from G to A). Therefore, the cost will be low.



Figure 3

The horizontal cost is defined to be the average transition cost taking into account all pairs of notes consisting of one note from each chord. In the example above, the horizontal cost equals  $(H\text{Cost}(C \rightarrow F) + H\text{Cost}(C \rightarrow A) + H\text{Cost}(E \rightarrow F) + H\text{Cost}(E \rightarrow A) + H\text{Cost}(G \rightarrow F) + H\text{Cost}(G \rightarrow A)) / 6$ .

We determine the horizontal costs with a similar method as for the vertical costs. The situation is simple when fingers are in ascending order when we are playing notes from lower to higher pitch or descending order in the opposite direction. In this case the cost is calculated as if the two notes were played simultaneously using the vertical cost function. Otherwise, a crossover has occurred and different costs are involved. We partition the crossover situation into two categories:

- 1- Crossovers with the first finger being the thumb
- 2- Crossovers involving all fingers except the thumb

Situations in the first category are assigned relatively low costs as they are natural for the hand, while the second category is heavily penalized.

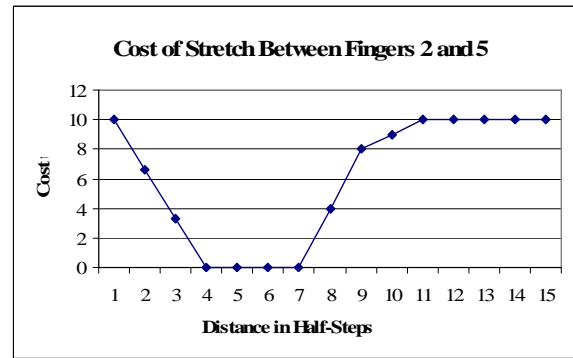


Figure 4

#### 4.2 Cost Parameter Initialization

For each pair of fingers, a cost function is defined that describes the difficulty of stretching that pair of fingers to span different distances. Each function is initialized to the same general shape as the example in Figure 4, which represents all possible costs of stretches involving fingers 2 and 5. The x-axis of the function is the distance in half steps between the notes played by these two fingers. The y-axis ranges between 0 (lowest cost) and 10 (highest cost). Distances greater than 15 half steps are defined to have a cost of 10.

Each function always has this general shape, although three of the inflection points (at distances 3, 7, and 9 in the figure) depend on the particular pair of fingers under consideration. These inflection points are determined by asking the user a series of questions to calibrate the system. In this example, the user has specified the following information:

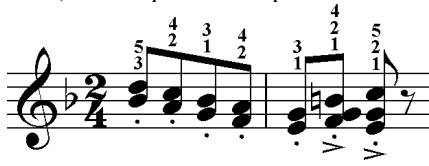
- The smallest interval that is easily playable by fingers 2 and 5 is 4 half steps.
- The comfort zone lies between 4 and 7 half steps.
- The difficulty continually increases between 7 and 11 half steps.
- After 11 half steps the stretch is nearly impossible.

After initialization, machine-learning techniques (described in a later section) are used to refine the parameters of the cost functions.

#### 4.3 Dynamic Programming

We can generate a fingering for a new input piece using a dynamic programming approach that minimizes the sum of the vertical and horizontal costs of all the chords of the piece. The problem consists of choosing the best fingering possible for each note of the piece, which reduces to choosing the best sequence of hand positions among the possible arrangements. Therefore, it is conveniently solved using a trellis graph and dynamic programming to find the path with the smallest total cost between the first and last chords of the piece.

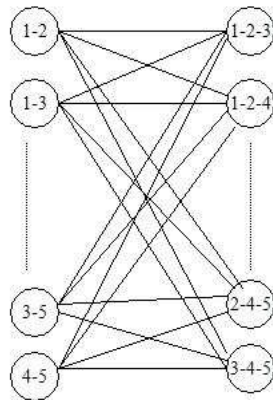
Each layer of the trellis graph contains a set of nodes representing the different possible hand positions (fingered chords) at each point in the piece.



**Figure 5.** Excerpt from Spinning Song by Albert Ellmenreich

The trellis graph of the excerpt above will have 7 layers, each layer corresponding to a played chord. Figure 6 shows part of the fifth and sixth layers of the trellis graph. The 5 first layers will each contain 10 vertices representing the possible combinations of two fingers (1-2, 1-3, 1-4, 1-5, 2-3, 2-4, 2-5, 3-4, 3-5 and 4-5). The two last layers will also each contain 10 nodes that correspond to all possible combinations of three fingers (1-2-3, 1-2-4, 1-2-5, 1-3-4, 1-3-5, 1-4-5, 2-3-4, 2-3-5, 2-4-5 and 3-4-5).

Each layer is fully connected to the next layer. A single “start” node is connected to each node in the first layer, with costs simply set to the vertical cost of each vertex in the layer. The nodes of the last layer are joined to one single “end” node with weights equal to 0.



**Figure 6**

The cost of transitioning from node  $n_i$  in layer  $l_i$  to node  $n_f$  in layer  $l_f$  in the graph is a sum of :

- The vertical cost for node  $n_f$  in layer  $l_f$
- The horizontal cost of moving from node  $n_i$  to  $n_f$ .

The last step is to find the shortest path between the “start” node and the “end” node using Dijkstra’s algorithm. The path with the best score (lowest overall cost) will be chosen as optimal fingering sequence.

### 3 MACHINE LEARNING OF PARAMETERS

The fingering chosen by the system for a piece is completely determined by the shape of the cost functions defined for each pair of fingers. To improve system performance, we provide training examples consisting of MIDI files and associated fingering files. The total error for a fingering generated by the system is calculated by counting the number of notes that receive a different finger number than expected. We use an iterative algorithm similar to simulated annealing to adjust the cost functions in response to errors. This is a credit assignment problem in that we must indirectly infer which specific factors led to an incorrect fingering after the entire dynamic programming algorithm produces its output.

Credit assignment is quite difficult in general, so this algorithm makes the simplifying assumption that errors are caused by *local* miscalculations of true cost. To illustrate the algorithm, suppose that the G in the second-to-last chord in Figure 5 had been fingered incorrectly by the system, yielding a chord fingering of 1,3,4 instead of 1,2,4. Cost function parameters would only be adjusted for those functions describing the following local note pairs:

- Vertical pairs: pairs involving adjacent notes in the chord containing the error. In the example, the costs of the stretches from finger 2 to 4 and from finger 1 to 2 are candidates for correction via reduction in cost. Similarly, the costs from stretches from 1 to 3 and 3 to 4 produced in the system output are candidates for increase in order to penalize this choice.
- Horizontal pairs: pairs involving the note that was misfingered and the one or two notes closest to it in the preceding and following chords. Here, the only pairs involved are transitions between different fingers on note G.

The algorithm (implementation in-progress) proceeds as follows:

1. Generate fingerings for the training data.
2. Calculate the total error in the output.
3. For each incorrectly fingered note:
  - a. Identify the pairs of fingers potentially involved in generating the error.
  - b. Ignore any identified pairs where the distance between fingers is in the “comfort zone”.
  - c. Modify the relevant cost functions slightly to lower the cost for the pairs involved in the target fingering. Raise the cost for the pairs involved in the incorrect generated fingering.
4. Return to step 1 unless the error is below a specified threshold or the maximum number of training epochs has been exceeded

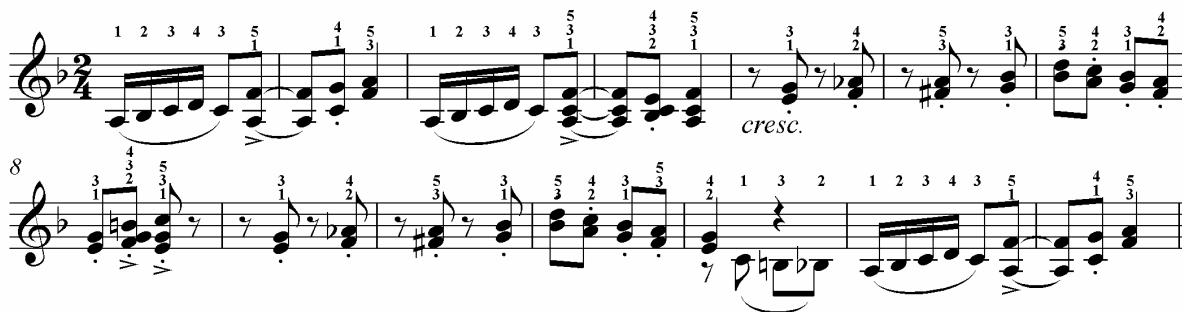


Figure 6. System output: Excerpt from Ellmenreich's Spinning Song

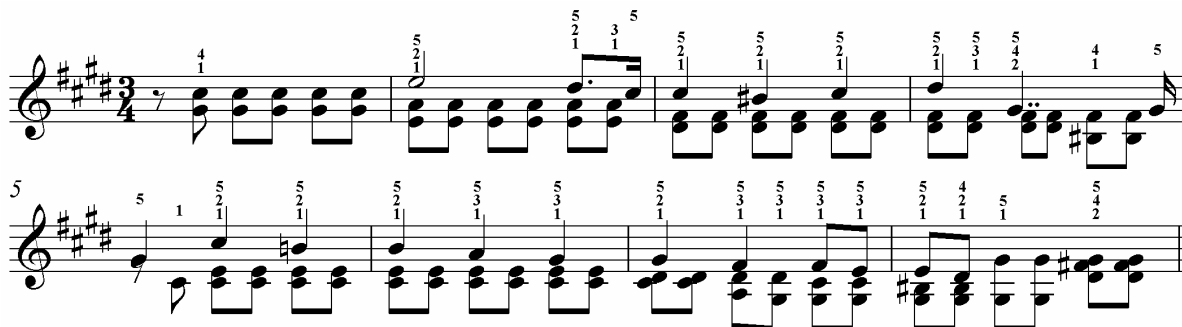


Figure 7. System output: Excerpt from Chopin's Etude op. 25, no. 7

#### 4 RESULTS

We tried our system on several different music excerpts and the preliminary results were very encouraging. The fingering output in Figure 6 features very smooth transitions. Chords were given appropriate stretches between fingers, and stepwise passages such as in the first measure were given realistic fingerings that would normally be used by a pianist. However, a surprising fingering occurs in measures 3-4 on the A-C-F and Bb-C-E chords. Here, the choice of 1-3-5 and 2-3-4 occurred because the importance of smooth horizontal transitions from 1 to 2 and from 5 to 4 outweighed the cost of the awkward vertical hand stretch. A more natural fingering would be 1-2-5 and 1-2-4, which involves playing two adjacent notes (Bb and C) with the same finger (1). However, this situation was slightly penalized in our model compared to other transitions.

We can see similar situations in the second excerpt (Figure 7), where the horizontal and vertical weights are out-of-balance. To improve system output we are currently refining the shape of the cost functions used to

determine the horizontal and vertical costs. Likewise, the machine learning system will use training data to enhance the results by improving the weights assigned for each specific finger transition.

#### ACKNOWLEDGEMENTS

The authors would like to express thanks to Prof. David Leake for his assistance in the machine learning concepts.

#### REFERENCES

- [1] Mitchell, T. *Machine Learning*. McGraw-Hill, 1997.
- [2] Radicioni et. al. "A segmentation-based prototype to compute string instruments fingering". *Proceedings of the Conference of Interdisciplinary Musicology (CIM04)*. Graz/Austria, April 15-18, 2004. URL: <http://gewi.uni-graz.at/~cim04/>
- [3] Sayegh, S. "Fingering for String Instruments with the Optimal Path Paradigm". *Computer Music Journal*. Vol. 13, No. 3, Fall 1989, MIT.