A Differential Evolution Algorithm Assisted by ANFIS for Music Fingering

Roberto De Prisco, Gianluca Zaccagnino, and Rocco Zaccagnino

Musimathics Laboratory Dipartimento di Informatica
Università di Salerno 84084 Fisciano (SA) - Italy
http://music.dia.unisa.it

Abstract. Music fingering is a cognitive process whose goal is to map each note of a music score to a *fingering* on some instrument. A *fingering* specifies the fingers of the hands that the player should use to play the notes. This problem arises for many instruments and it can be quite different from instrument to instrument; guitar fingering, for example, is different from piano fingering. Previous work focuses on specific instruments, in particular the guitar, and evolutionary algorithms have been used.

In this paper, we propose a differential evolution (DE) algorithm designed for general music fingering (any kind of music instruments). The algorithm uses an Adaptive Neuro-Fuzzy Inference System (ANFIS) engine that learns the fingering from music already fingered.

The algorithm follows the basic DE strategy but exploits also some customizations specific to the fingering problem. We have implemented the DE algorithm in Java and we have used the ANFIS network in Matlab. The two systems communicate by using the MatlabControl library. Several tests have been performed to evaluate its efficacy.

1 Introduction

Given a music score for some instrument, a *fingering* is a mapping of each note of the input score to a specific "position" of the hands that should be used to play the notes. The position of the hands in most cases just specifies a finger for each note (like for example for piano scores). In some other case it specifies also other information (like a string on the guitar). In some cases also the two foots are used (like for the church organ). A *fingered* music score is a music score with a fingering.

Fingering involves several aspects: musical analysis, physical constraints, biomechanical constraints (possible figures of the hand). In addition, each musician has different preferences about the positioning of the fingers, as suggested by experience, physical possibilities and so on. Fingered music can be of great help to music students or any one that wishes to play and does not have enough competence to find a good fingering by himself. The process of fingering a music score, however, can be laborious and time consuming, especially if one has plenty of music to be fingered. A publishing house might want to print a book of music with the fingering rather than without it. Having computer programs that can automatically find music fingerings can be of great help. A music score can have many possible fingerings. Although each instrument has specific constraint that might reduce the number of possible fingerings, in theory, there is a huge number of possible fingerings for a music score. This makes the evolutionary approach to this problem interesting.

The fingering problem has been given considerable attention in the last few years, although most of the work focuses on the guitar. Each instruments has specific physical and structural features which make the fingering problem instrument-dependant.

In this paper we explore the use of the Differential Evolution (DE) approach for the fingering problem. Although our algorithm follows the general DE strategy, there are several customization specific to the fingering problem. In our approach, the DE algorithm is used to explore good fingered configurations among the solution space. Once the algorithm generates a new solution, an adaptive neuro-fuzzy inference system (ANFIS) [5] is used to determine its fitness value for the evolutionary algorithm to continue its search process. ANFIS is a class of adaptive networks which are functionally equivalent to fuzzy inference systems. In our case, the ANFIS network is trained to learn fingered positions starting from music already fingered. Such fingered music might represent either the particular preferences of a specific user (musician) or standard fingering practice for the instrument for which the music is written.

This paper. In this paper, we propose a general model of fingering, not restricted to one specific instrument but usable for any type of musical instrument, although the algorithm uses a different representation of the fingering which is instrument-dependant. This is transparent to the user. In order to abstract from one specific instrument, the model does not use information on the physical characteristics of the musical instruments, but gets the needed information from already fingered music. To the best of our knowledge this is the first algorithm applicable to any type of musical instrument. We have implemented the DE algorithm in Java and the ANFIS network in Matlab. The two systems communicate by using the MatlabControl library. We have run several tests and finally, the output of the system is validated against the performance of a human expert. In the final section of the paper we report the results of the tests.

Related work. Most of previous works are concerned with the fingering of stringed instruments (in particular the guitar). Expert systems for the guitar fingering problem have been published by Sayegh [10], Miura and Yanagida (MY) [6] and Emura et al [3]. Radisavljevic and Driessen [9] implement and build on Sayegh's idea of dynamic programming. Their interest is in tuning their algorithm to particular styles of music fingering through training over selected scores for guitar. Tuohy and Potter [13] approach the guitar fingering problem with a genetic algorithm. In this algorithm the population is a collection of tablatures that are valid for a given piece of music. A tablature "chromosome" is defined as a sequence of chords. A chord is a "gene" and consists of fretboard positions for all the notes in that chord. The fitness function is based on two separate classes of tablature complexity: difficulty of hand/finger movement and difficulty of hand/finger manipulation. In a later work [14] they introduce a neural network to assign fingers to their tablature.

2 Background

2.1 Music Notation

We assume that the reader is familiar with basic music notions. The twelve notes of an *octave* are denoted with the letters A, A♯ or B♭, B, C, C♯ or D♭, D, D♯ or E♭, E, F, F♯ or G♭, G and G♯ or A♭. In the audible range of sounds there are several octaves, usually denoted with the numbers from 0 to 7. The keyboard of a piano contains all the 88 notes used in music (the lowest notes and also the highest ones are used rarely). We will use MIDI codes to specify the notes. Each MIDI code is a number in the range 0-127. Thus we have more MIDI codes than keys in a standard 88-key piano. The lowest note in the piano, the first A, has MIDI code 21, while the highest note, the last C, has MIDI code 108.

Fingering information is instrument-specific, Figure 1 shows an example of fingering for piano and Figure 2 shows an example of fingering for guitar.

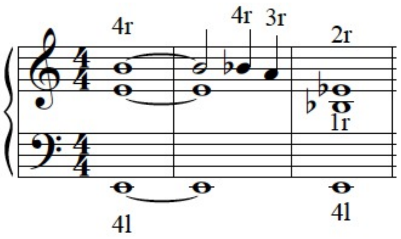


Fig. 1. Piano fingering

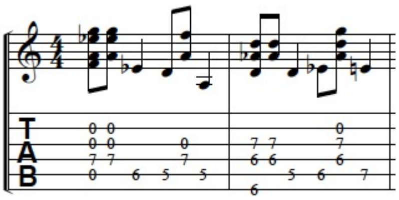


Fig. 2. Guitar fingering

Given a score S it is possible to have many fingerings $F_i(S)$. Although each instrument has specific constraint that might reduce the number of possible fingerings, in theory, given a score S there is a huge number of fingerings for S. The goal of the algorithm is to find a "good" fingering, one that would be used by an expert musician.

2.2 Differential Evolution

Given the scope of the conference we assume that the reader is familiar with evolutionary algorithms and in particular with the differential evolution (DE) strategy proposed in [11]. In the next section we will describe the modifications needed to adapt the standard (DE) strategy to our problem.

2.3 Adaptive-Network-Based Fuzzy Inference System (ANFIS)

An *Adaptive Network-Based Fuzzy Inference System* or simply *ANFIS* can be used for constructing a set of fuzzy if-then-rules with appropriate membership functions able to generate correct input-output pairs. Due to lack of space we refer the reader, for example, to [4,12] for more information. In the next section we will explain how we use an ANFIS network for our algorithm.

3 The DE Algorithm

In this section we present the fingering algorithm that we call DE. The DE algorithm adopts the differential evolution strategy and uses an ANFIS network to evaluate the fitness of the solutions. To describe the algorithm we start by describing how we represent individuals, then we describe how we use the ANFIS network and finally how we adapt the DE strategy to the fingering problem.

3.1 Data Representation

In this section we describe how we represent a fingering. The choice of the data representation is a crucial step in the design of an evolutionary algorithm and also of a fuzzy network. We seek a representation that is enough general to deal with fingerings for all kind of instruments.

Regardless of the specific instrument for which it is written, a music score S can be viewed as a temporal sequence of *score-changes*, where each score-change is the appearance of a new note or group of notes. We denote this sequence as $S = S_1, S_2, \ldots, S_N$, where N is the number of score-change in S. Notes in a score-change are specified using the corresponding MIDI code. Figure 3 provides an example. The score fragment consists of 4 score-changes. Notice that the score-change representation of a score loses the information about timing (which is not needed for fingering) retaining only the information about the notes to be played.

Abstracting from the instruments a fingering configuration will be a 12-element array of *notes-information* and an associated 12-element array of *extra-information*. The notes-information will be MIDI codes that specify the score-change. We use a maximum of 12 notes since we assume that with the

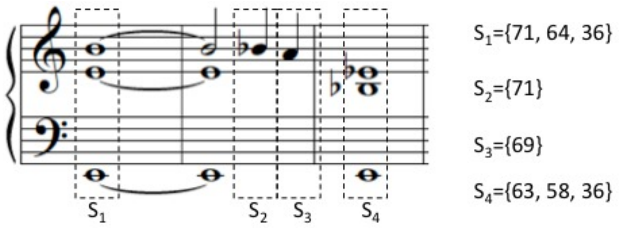
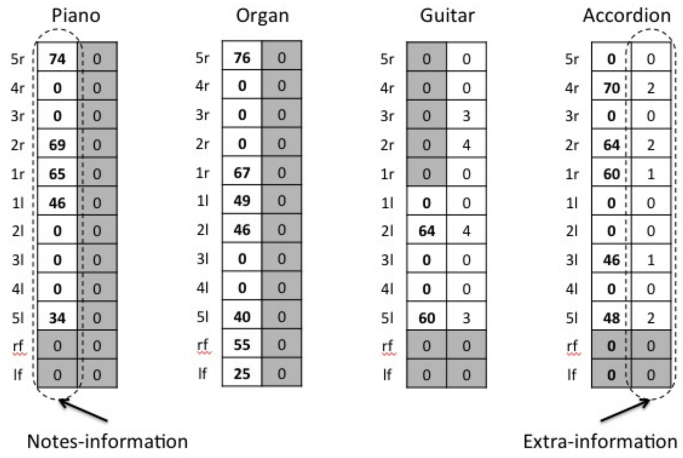


Fig. 3. A fragment of a score and the sequence of score-changes

fingers we can play at most 10 notes and on some instruments (like the organ) we can use the foots to play other 2 notes[1] For many instruments, in particular the ones having a keyboard, the notes-information is all we need to specify a fingering. For other instruments, like for example a guitar, we need additional information. In such cases we use them *extra-information.*

Figure 4 shows examples of fingering. Gray elements are not used. The fingering for piano and organ simply map each note to a finger, numbered from 1 (thumb) through 5 (little), or a foot (left or right). For the guitar we use the notes-information for the left hand: in the example the 2^{nd} and the 5^{th} finger play E (MIDI code 64) and C (MIDI code 60). Moreover the extra information for the left hand tells on which string the fingers should be placed: the 2^{nd} finger on the 4^{th} string and the 5^{th} finger on the 3^{rd} string. Finally the extra-information for the right hand tells which finger of the right hand should pluck the strings specified in the extra-information of the left hand: the 3^{rd} finger should pluck the 3^{rd} string and the 2^{nd} finger the 4^{th} string. For the accordion the extra-information specifies the row of buttons to be used (some notes can be played in different rows).



	Piano		Organ		Guitar		Accordion	
5r	74	0	76	0	0	0	0	0
4r	0	0	0	0	0	0	70	2
3r	0	0	0	0	0	3	0	0
2r	69	0	0	0	0	4	64	2
1r	65	0	67	0	0	0	60	1
1l	46	0	49	0	0	0	0	0
2l	0	0	46	0	64	4	0	0
3l	0	0	0	0	0	0	46	1
4l	0	0	0	0	0	0	0	0
5l	34	0	40	0	60	3	48	2
rf	0	0	55	0	0	0	0	0
lf	0	0	25	0	0	0	0	0

Fig. 4. A fingered score fragment and its matrix representation

Although we have chosen to represent the fingering as a dimensional matrix (just because we have notes-information and associated extra-information), we can store all the fingering information for a score change in one single array of dimension K. The implementation uses $K = 24$ although, depending on the instrument, some entries are always 0.

[1] Actually on the organ it is possible to play more than 1 note with each foot. For simplicity we assumed that one can play only one note. In order to accomodate more than one note per foot it is enough to make the array longer.

Given a score S, an entire fingering $F(S)$ is represented as a $K \times N$ matrix, where N is the total number of score-changes in S, and each column represents one fingering-change.

3.2 The ANFIS Model for Learning Fingering

The objective of this network is to learn the preferred fingering, be it that of a single-user musician or that deriving from common practice. The choice of a musician on how to place the fingers for a specific score-change depends on the fingering for the previous score-change and that of the next score-changes. This is because each player tries to move his hands in the most convenient way and as efficiently as possible. Thus the ANFIS network will work with *triples* of consecutive fingering-changes Each instance in the training set is a pair (*triple of fingering-changes, preference*).

To represent each such a pair, we need an array of $3K + 1$ elements, the first $3K$ to represent the three fingering-changes (each one takes K integers), while the last entry of the array will contain the preference that the user gives to this particular fingering.

Let $\{S^1, \ldots, S^M\}$ be a set of scores set and $\{F(S^1), \ldots, F(S^M)\}$ be the corresponding fingerings. We consider all the possible the triples of score-changes $S^i_{j-1}S^i_jS^i_{j+1}$ for all $i = 1, \ldots, M$ and all $j = 2, \ldots, N_i - 1$, where N_i is the number of score changes in S^i (remember that each score-change is a set of notes). Then we count the number of occurrences of a particular fingering for each triple. Such a number (normalized by the total number of appearance of the triple of score-changes to which the fingering is referred) gives the preference for that particular fingering.

3.3 The Differential Evolution Algorithm

The DE algorithm takes as input a score and produces a fingering for the score.

Encoding. The population in our algorithm is made up of individuals that are fingered configurations of the given input score. We represent each individual using the data representation explained in previous sections; that is a chromosome x (individual) is a fingering $x = F(S)$ for the input score S and each fingering-change is a gene of x.

Fitness Measure. The *fitness value* $f(x)$ is defined as follows. Let x be an individual of the population, and $F(S)$ the corresponding fingering. For each $i = 2, \ldots, N - 1$, we consider the triple of consecutive genes $\langle F_{i-1}, F_i, F^{i+1} \rangle$ of $x = F(S)$ and we use the neuro-fuzzy network to evaluate the triple. Let $f(i)$ be the value returned by the neuro-fuzzy network. The overall evaluation of x is

$$f(x) = f(F(S)) = \sum_{i=2}^{N-1} f(i).$$

While evaluating chromosomes our algorithm computes also important information. In particular for each individual we identify points that we call *cut points.*

The cut point is the triple of consecutive genes (score-changes) that contains the worst value. We used a similar approach in [1,2], for a different problem. In the rest of the paper we will identify cut point i with the triple of genes $F_{i-1}F_iF_{i+1}$ that gives the minimum value over all possible i.

Initial Population. We start with a random initial population. We build chromosomes by selecting a random values for entry of the fingering configuration.

Mutation (perturbation) Operator. Now we have to specify how we compute the perturbation. Since the individuals of our problem cannot be described as real numbers over a continuous domain we have to introduce different ways to define the "differential" mutation. To do so we will define specialized '+' (addition) and '×' (multiplication) operations that work with the individuals of our problem.

Let F^1, F^2, F^3 be the three parents chosen at random. We have to compute the mutant vector. To do so, we need to introduce the '+' and '×' operation for the individuals of our problem.

The mutant vector is given by $v = F^1 + Z \times (F^2 - F^3)$, where Z is the scale factor. The "difference" F^2 and F^3 between two fingering is a vector of integers values and is computed as follows. Let $S_i = \{a_1, a_2, \ldots, a_k\}$ be the notes in the i^{th} score-change. For each note $a_j, j = 1, \ldots, k$, the j^{th} component of the difference for the notes-information D_j is given by the change of position of note a_j from F^2 to F^3. For example if note a_j is played by the 4^{th} finger of the right hand in F^2 while in F^3 is played with the 5^{th} finger of the right hand, then $D_j = -1$. For the extra-information, the difference is defined in a similar way (the exact definition depends on the instrument).

To increase the perturbation we use the crossover operator. The crossover operator starts from the mutant and the target vector and produces the trial vector. Let $v = (M_1, ..., M_N)$ be the mutant vector. The target vector is x_i. Let $x_i = (G_1, ..., G^N)$. Moreover let the trial vector be $u = (T_1, ..., T_N)$.

We augment the basic DE algorithm by using two possible crossover:

1. Binomial Crossover. That is, the trial vector is obtained as:

$$T_j = \begin{cases} M_j & if(rand([0,1]) \leq \text{CR} \;\; or \;\; j = i \\ G_j & otherwise \end{cases}$$

   where CR is a user-defined crossover threshold in the range [0, 1] and *rand(j)* is the j^{th} evaluation of a uniform random number generator that returns numbers in the range [0, 1].
2. Cut crossover. In this case we exploit the cut point to perform a crossover operation. Let k be the cut point in the target vector x_i. Then the trial vector u is obtained considering the two individuals $F^1 = (G^1, ..., G^k, M^{k+1}, ..., M^N)$ and $F^2 = (M^1, ..., M^k, G^{k+1}, ..., G^N)$. Finally the trial vector u is simply the individual F_z, $z = \{1, 2\}$ with the best fitness value.

To choose which crossover we have to use we exploit a probability distribution $\mathcal{P}_C = (p_n, p_c)$, where the specific choice for the two probabilities can be tuned to improve the results of the tests.

4 Test Results

In order to evaluate the proposed system we have run several tests. As test cases we have used three type of instruments: the piano, the guitar and the chromatic accordion.

4.1 ANFIS Model: Training, Checking and Test Results

Our ANFIS model for fingering preferences is developed using the Matlab Fuzzy Logic Toolbox. For each instrument we selected a set of already fingered scores from which we extracted a data set of triples of fingering changes for each instrument and we used them as shown in Table 1.

Table 1. Number of data instances and performance measures of the ANFIS network

	total instances	training instances	checking instances	testing instances	learning epochs	minimum MSE	coefficient of determination
Piano	1100	770	115	115	110	0.112	0.8722
Guitar	1280	896	192	192	132	0.124	0.8018
Accordion	1030	721	155	154	147	0.132	0.8671

The ANFIS network has been created using generalized bell membership functions with five parameters. To evaluate the errors we used the *MSE* (Mean Square Error) function. We trained the network using the hybrid learning algorithm over a maximum of 200 epochs. For the piano, the total sum of the MES converged to a minimum of 0.112 after 110 training epochs. Therefore for the final learning we used 110 epochs. See Table 1 for the other instruments.

After training we validated the ANFIS network against the testing instances. The coefficient of determination, a number in $[0, 1]$ which specifies the goodness of the network, with 1 being the best possible result, is shown in Table 1.

4.2 Test Results and Conclusions

We have run several tests varying the size of the initial population and the number of generations. The other parameters involved in the tests are the scale factor Z, the crossover threshold CR and the probability distribution $\mathcal{P}_C$ for the crossover operation. We have used $Z = 0.7$ and $CR = 0.8$ as suggested in [7]. For $\mathcal{P}_C$ We have tried several choices and the one that gave best results is $\mathcal{P}_C = \{\frac{1}{3}, \frac{2}{3}\}$.

Finally we have asked an expert musician to evaluate the output of the DE algorithm. We have considered two pieces for piano from the standard jazz repertoire, *Round Midnight* by Thelonious Monk and *All Blues* by Miles Davis, two pieces for guitar from the standard Latin jazz repertoire, *Wave* and *How Insensitive* by Jobim, two pieces for accordion from the tango repertoire, *Libertango* by Astor Piazzolla and *A Evaristo Carriego* by Eduardo Rovira.

For each instrument, we have requested the musician tell which fingerings were correct and which ones were wrong. The percentage of correct fingerings was 89.2% for the piano, 87.3% for the guitar and 85.3% for the accordion.

Finally we have compared the DE algorithm with a standard genetic algorithm (GA) with same features, that is using the same chromosome representation, the same fitness evaluation function, the same ANFIS network. The DE algorithm always outperforms the genetic algorithm. Future work include the investigation of DE variants (see for example [8]).

References

1. De Prisco, R., Zaccagnino, G., Zaccagnino, R.: EvoBassComposer: a multi-objective genetic algorithm for 4-voice compositions. In: Proceedings of the 12th ACM Annual Conference on Genetic and Evolutionary Computation, GECCO 2010, pp. 817–818 (2010)
2. De Prisco, R., Zaccagnino, G., Zaccagnino, R.: A multi-objective differential evolution algorithm for 4-voice compositions. In: Proceedings of IEEE Symposium on Differential Evolution, SDE 2011, pp. 817–818 (2011)
3. Emura, N., Miura, M., Hama, N., Yanagida, M.: A system giving the optimal chord-form sequence for playing a guitar. Acoustical Science and Technology (January 2006)
4. Kandel, A.: Fuzzy expert systems. CRC Press, Boca Raton (1992)
5. Jang, J.R.: ANFIS: An Adaptive-Nework-Based Fuzzy Inference System. IEEE Transactions on Systems, Man, and Cybernetics 23(3) (May/June 1993)
6. Miura, M., Yanagida, M.: Finger-position determination and tablature generation for novice guitar players. In: Proceedings of the 7th International Conference on Music Perception and Cognition (2002)
7. Neri, F., Tirronen, V.: Scale factor local search in differential evolution. Memetic Comp. 1, 153–171 (2009)
8. Neri, F., Tirronen, V.: Recent Advances in Differential Evolution: A Review and Experimental Analysis. Artificial Intelligence Review 33(1), 61–106
9. Radisavljevic, A., Driessen, P.: Path difference learning for guitar fingering problem. In: Proceedings of the International Computer Music Conference (2004)
10. Sayegh, S.: Fingering for string instruments with the optimum path paradigm. Computer Music Journal 6(13), 76–84 (1989)
11. Storn, R., Price, K.: Differential evolution: a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report TR-95-012, International Computer Science Institute, Berkeley
12. Takagi, T., Sugeno, M.: Fuzzy identification of systems and its applications to modeling and control. Proceedings of IEEE Transactions on Systems, Man, and Cybernetics 15, 116–132 (1985)
13. Tuohy, D., Potter, W.: A genetic algorithm for the automatic generation of playable guitar tablature. In: Proceedings of the International Computer Music Conference (2004)
14. Tuohy, D., Potter, W.: GA-based music arranging for guitar. In: Procedings of IEEE Congress on Evolutionary Computation, CEC 2006 (2006)