An Idiomatic Plucked String Player

Leandro L. Costalonga[1], Eduardo R. Miranda[1], John Matthias[1], Rosa M. Vicari [2]

1)Interdisciplinary Centre for Computer Music Research
Faculty of Techonology, University of Plymouth
Plymouth, Devon PL4 8AA, United Kingdom
{leandro.costalonga; eduardo.miranda;
john.matthias}@plymouth.ac.uk

2)Universidade Federal do Rio Grande do Sul (UFRGS)
Instituto de Informática
Porto Alegre – RS – Brazil. PO.Box 15.064
rosa@inf.ufrgs.br

Abstract

We are developing systems which play music on synthesized plucked strings instruments idiomatically. The present paper introduces a system focusing on the acoustic guitar. The system is programmed with rules embodying the architecture of the guitar and the respective biomechanical constraints of a performer. These rules define the types of musical gestures that are played by the system: only those gestures that are idiomatically compatible with the guitar are performed. The system also features rules for arranging (and re-arranging) musical materials in order to make them idiomatically playable. These include rules for shaping chords and algorithms for generating realistic fingering.

Introduction

There are a great number of sound synthesis techniques and computer-aided composition systems available. However, there is a general lack of intelligent interfacing between these two types of systems. For example, whereas systems that generate music for guitar using Cellular Automata (Miranda, 1993) may produce interesting musical passages, they often sound rather uninteresting when played back on a synthesized guitar. This is due to the fact that these systems do not "know" anything about the guitar itself. And indeed, such passages could have been played using any of the timbres of a MIDI synthesis module and would have sounded equally unsatisfactory. The same problem applies to software for musical notation. For example, when one listens to the different parts of an orchestral piece, the software treats these materials identically; that is, the violin part is not played "violinistically" or the clarinet part "clarinetistically".

Musicians tend to blame the synthesizers for such poor performance quality, but in fact the problem is an idiomatic problem. If these systems knew how to control a synthesizer idiomatically, the quality of the synthesis itself would not matter so much.

The idiomatic problem is not, however, only limited to the lack of knowledge about the instruments in question. It also includes knowledge about the biophysical constraints of the performer, the performance characteristics associated with different musical styles and the performance styles of different individuals.

The aim of this research is to improve this scenario by adding intelligence to the interface between systems for computer-aided composition and synthesizers.

In this paper we introduce a system that is able to play guitar music on a synthesized guitar idiomatically. It is furnished with knowledge about the architecture of the guitar and the biophysical constraints of the performer (e.g., fingering constraints).

Note, however, that the system presented here goes beyond the notion of an intelligent MIDI playback device: it is also is an arranger. The system takes a sequence of chords (in guitar chord text notation) and rhythm patterns, and then it arranges the music and interprets it. This scenario is inspired by the fact that pop music is usually notated using chord text notation or tablature (tab) notation. Interpreters then define the shape of the chords, fingering, rhythms, etc. according to the style of the piece in question.

Music Interpretation

Musicians have their own performance styles and interpretations of musical structures, resulting in high degrees of deviation from the notation of the score (Zanon and DePoli, 2003). Patterns of deviations were presented by Repp when he analyzed the timing and articulation of many pianists. These deviations communicate their expressive intentions and emotions (Repp, 1990). Most studies of music performance use the word "expressiveness" to indicate the systematic presence of deviations. (Gabrielsoon, 1997).

AI methods were used to study the complex phenomenon of music performance with the goal of discovering simple, general and interpretable models of aspects of expressive music performance (such as timing, tempo, dynamic (loudness variation) and articulation (the way how successive notes are connected) (Widmer, 2003).

Neural network and genetic algorithms have been used to interactively generate rhythm patterns to the user's general stylistic requirements (Burton and Vladimirova,

1997). Other pattern discovery algorithms such as SIATEC (Meredith et al., 2001) and FlExPat (Rolland, 2001) can be used to train the neural networks. Hörn and Menzel presented neural network models able to learn the structural elements (harmony and melody) that characterize a composer's personal style (Hörnel and Menzel, 1998).

Werner Goebl proposed a Self-Organization Map algorithm fed by six famous pianists playing six Chopin pieces, where the researchers can control the type of normalization, the degree of smoothing and the weighting between tempo and volume. Although they obtained interesting results, the algorithm did not consider the volume of individual notes (Goebl, 2004).

Our work differs from previous work in that we are also considering the chord shapes used to play the piece and, in addition, how the guitarist actuates the transitions between them. This is similar to the articulation concept proposed by Widmer, but in a guitar context.

The Guitar and Guitarist's Constraints

String-fretted instruments have many peculiarities in the manner in which they are played. However, before considering the way humans play the guitar, we should think about the instrument's characteristics that are relevant to our system. They are:

Number of strings: Usually, a regular guitar has 6 strings. Other string-fretted instruments have different settings, but normally this number stays between 4 and 12.

Tuning: The regular guitar tuning is E(low) A D G B E(high), from the 6th to the 1st string. Other instruments may have different tunings, and even the guitar may have its tuning changed to play specific styles of music or pieces. For example: D A D G B E was used in the early 1990's to play chords more loudly with less effort in Nirvana's style. The D A D G A D tuning was explored by Jimmy Page and other rock guitarists. Many Rolling Stones riffs were played by Keith Richards using D G D G B D.

Number of frets: Frets indicate fractions of the length of a string and consequently the note. The number of frets varies according to the style and model of the guitars; usually the number stays between 19 and 24, even though we can reduce the number of frets using the guitar capo/cheater.

It is not just the instrument constraints which are important. It is also important how the musician deals with them. In the context of our work, which assumes a right-handed guitarist, the constraints are:

Number of left-hand fingers: The classical technique requires the use 4 fingers to execute the chords. Some guitarists also use the thumb in the upper strings, which means, they can utilize 5 fingers. Also, for some reasons such as hand imperfections, a guitar player may have fewer fingers available to execute the chords.

Number of right-hand fingers: Once again, the classical technique requires the use of 4 fingers from the right hand. However, some musical styles, such as the Spanish flamenco, have rhythm particularities, which can use more than 4 fingers or use them for tapping, muffling, up-stroking, and down-stroking and so on, in addition to picking the strings. The use of picks/plectrums in electric guitars is very common, which is equivalent to using 1 finger (but much faster). This is directly related to the way the guitarist execute the chord.

Finger Extension: A chord shape that requires a large extension of the fingers cannot be easily executed by beginners. This extension normally increases with practice. Usually, in the guitar, a 4th finger extension is desirable.

Skill level: Beginners normally avoid complex or unknown chord shapes. Barre chord shapes can also be difficult to execute, so the system can adapt the suggested chord's shape to make it more idiomatic. Complexity of the chord execution will be treated later.

Chord Textual Notation Recognition

In informal song writing or accompaniment situations, guitarists usually write their songs just using the chord's textual notation or an execution notation such as the tablature.

Therefore the first problem to be treated by the system is the recognition of the chords given to the system. The textual notation that is used to represent chords is not completely standardized. The same set of notes can be written in different ways. Take, for example: C, E, G, B. These notes can represent either a C7M or an Em/C chord. Moreover, C7M chord can be written as Cmaj7. The way some chords' symbols are used is contradictory, making a computational formalization even more difficult. To solve this problem, we have used finite automata. The default language symbols were defined based on the most common chord text notation used in Brazil (as seen in Table 1.), but it can be customized or even replaced whenever the developer or the user desires (even in runtime).

Table 1: Brazilian Chord Notation

Symbol	Description	Symbol	Description
A	Note A	dim	Diminished
B	Note B	2	Major 2nd
C	Note C	b2	Minor 2nd
D	Note D	4	Perfect 4th
E	Note E	#4	Augmented 4th
F	Note F	5	Perfect 5th
G	Note G	#5	Augmented 5th
#	Sharp	b5	Minor 5th
b	Flat	6	Major 6th
add	Interval addition	7	Minor 7th
(	Begin of a note alteration	7m	Major 7th
)	End of a note alteration	9	Ninth
/	Chord inversion	b9	Minor 9th
^	Interval junction	11	Major 11th
M	Major	#11	Augmented 11th
m	Minor	13	Major 13th
sus	suspensión		

Every symbol of the notation can be changed, but there are some fixed rules to be considered. The parenthesis (or any symbol for note alteration) is used when there is an alteration (# or b) in the basic intervals of the chord or when there is an interval that does not compose the basic structure of the chord (9,11,13). Also, it was established that only tetrachords could be considered diminished (1+b3+b5+b7). Equivalent triad must be 1+b3+b5, i.e. Cm(b5).

The "add" symbol is used when there is an additional interval that is not the next natural interval (related to the last one). E.g. C = 1+3+5; C(add 9) = 1+3+5+9. The next natural interval in this case should be the 7th.

The " ^ " non-musical symbol was created because of technical restrictions of the automata, once it considers the blank space as the end of the text word. If we have to validate a chord like C7M(9 11) we must replace the blank space by the "interval junction symbol".

After validation, the chord the system will mount the chord based on the chord formation rules (the notes will be attached to the intervals). An example of chord rule is: Minor (m) chord must have 1 + b3 + 5 intervals.

A special treatment was given to []chords where the bass note is different from the fundamental one, such as the C/G. This chord has an inversion, that is, the G note is part of the basic notes of the C chord (C + E + G), so it is just necessary to set the G note with the lower pitch. This is not the case with an Am/F# chord because the F# note is not part of the notes of the Am chord, so it must be added to the set of notes although the 6th interval is not represented in the chord symbols.

Chord Shape for Guitar

After identifying the chords, the system must discover how to play them. The chord shape defines how the fingers should be placed (fret and string) on the fretboard in order to produce the chord's notes.

There are many ways of playing the same chord. This is basically because most instruments, especially the harmonic ones, have a register larger than one octave, making possible the repetition of the same note. Chord notation does not determine the pitch of the notes, so this decision must be taken by the musician. In reality, several parameters interfere in this decision, starting with the user and the guitar constraints.

For computational performance reasons, the chord shape generation occurs in two phases. In the first phase, a set of simple chord shapes are calculated. Simple chord shapes are those with only one instance of the note suggesting that they can have some free strings to be used in the repetition of some notes (a chord cannot have more notes than the instrument's polyphonic capacity). The notes which will be doubled or repeated in addition with all the process related to the adaptation of the chord shape to the instrument or musician restrictions are made in the 2nd phase.

The decision to split chord shape calculus in two phases was made taking into account the processing time necessary to reach the best complete chord shape (with repeated notes). Without this strategy, all simple chord shapes must be processed and compared to find the best one, while working in phases we can compare only the best simple chord shape and process it. Chord shapes processed in the first phase can give a good idea about which chord shape to choose before slower processing. The function below returns the number of operations required to process each parameter given by *n*.

$$f(0) = 1$$

$$f(n) = n * f(n-1) + 1, \forall \{n \in \mathrm{N}^* \mid n \geq 1\}$$

Duplications, doubling, triplication and inversions are the processes made over simple chord shapes. It considers instrument restrictions (tuning, number of strings and frets) and user's profile (number of available fingers and maximum opening of fingers). The musician may set up the system in order not to calculate chord shape with, for example, the 3rd duplicated interval. When the system is processing the complete chord shapes it also suggests the right finger for each position (string +fret). The suppression is made only when it is not possible to calculate a simple chord shape, therefore during phase 1. The options are:

Fundamental note doubling: Repeats the fundamental note.

Fundamental note duplication: Repeats the fundamental note in unison.

Fundamental note triplication: Allows the repetition of fundamental note up to 3 times.

3rd interval doubling: Repeats the note related to the 3rd interval (major or minor).

Perfect 5th doubling: Repeats the note related to the 5th interval.

Perfect 5th duplication: Repeats the note related to the 5th interval in unison.

Perfect 5th suppression: Suppresses the 5th interval in order to add a high priority interval. This option is only considered when it is not possible to mount the chord shape with all the intervals defined in the chord symbols, so the 5th is omitted. This processing is part of phase 1.

Octave consideration (diatonic scale): Differs, for instance, a 2nd interval from the 9th, although they are the same note in different octaves. This is constantly ignored in guitar chords.

Fingering suggestion: Allows the system to suggest the right finger to each note position.

Barre chords: Allows the system calculate or not the barre chords (fretting).

Inversion calculus: Allows the system to calculate inversions in chord shape even if it is not expressed in chord symbols.

We consider for this work that duplications of notes occur in the same octave, while doubling can be in

different scales (Chediak, 1984). Therefore, duplication is a case of doubling.

Figure 1 shows how the processing occurs in a simple chord shape. In the example, 3 processes are set and they are represented by *p*1, *p*2 and *p*3. Observe that the simple chord shape "R" must tested for these 3 processes in the first level, the result tested by the other two processes in the second level and so on.

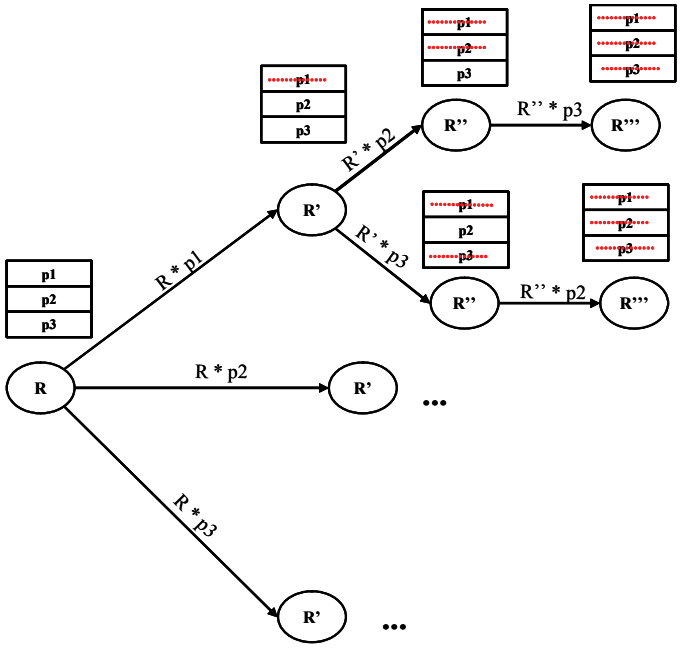

Figure 1: Processing made over a simple chord shape.

For a better understanding of the algorithm, suppose R is a C major with a simple chord shape given by 53(5th string, 3rd fret - C); 42(4th string, 2nd fret - E); 13(1st string, 3rd fret - G). Considering a regular 6-string guitar, we still have space to double notes in the 6th, 3rd, 2nd strings. What processes should be done are given by "p". So, p1 could be the "Fundamental note duplication" process and p2 a "Perfect 5th duplication". This means that all the combinations between available strings and doublings might be processed. So R' could be 53(C); 42(E); 21 (C); 13(G) – fundamental note duplicated. R' still have space 6th and 3rd strings, so the algorithm keeps searching for p2 and find R", what could be 53(C); 42(E); 30(G); 21 (C);13(G) – Perfect 5th duplicated, and so on. Not only we have to find the chords' shapes but also we have to eliminate possible repetitions, comparing them with the already know chord's shapes database.

The "Inversions calculus" option is considered when it is necessary to know if the search should start from the bass string of the instrument or from the bass note of the chord. Beyond this verification, it is necessary to verify if the musician will have enough fingers to play the chord and if the fretting is necessary.

The processed chord's shapes are sorted by: finger extension (decreasing), number of necessary fingers (increasing), and proximity to the guitar head (increasing). This process makes the first entry of the list the easiest chord shape to play, not necessarily the most appropriated or the one that sounds better. So the first chord of the music has a very important role in these decisions, because it can lead the chord shape to a very high region of the instrument, which is normally not used, although that chord shape can be easier to play. To solve this, the system chooses the chord shape that has the lower fret average, that is, the closest one to the guitar head.

Chords Shape Transitions/Fingering

At this point, the only criterion used to choose a simple chord shape is its similarity with the previous complete chord shape. This approach already shows some good results but must be improved. We plan to improve this by using a neural networks trained with recordings of human guitarists' performances.

The similarity factor is a value between 0 and 1, where 1 means the same chord. To find the most similar chord shape, all simple chord shapes are compared to the previous complete chord shape.

In string-fretted instruments there are some basic shapes that you can just move (maintaining the shape) to different frets to reach the desired chord. This "hand motion" is not considered by the similarity function, only the fingers motion.

To exemplify, suppose an Am chord shape being 50-42-32-21-10 followed by a G. The first step is to find all the simple chords shapes for G. Some of them are: 63- 40- 20; 63- 40- 34; 63- 52- 40; 510- 49- 110; 63- 34- 23; etc. Every simple chord shape for G will be compared with the complete chord shape for Am, generating a matrix as shown in Table 2. In this case, the selected chord shape for G was 63-40-20 with 52% of similarity.

Table 2: Am to G Transition

	-	**50**	**42**	**32**	**21**	**10**
63	-	.31				
40		.31	.62	.32		
20				.31	.62	.31

For every position (e.g. 63) of G, a value is given considering the positions of Am in the same string, one above and one below. For example, the 4th string open (40) of G will have a value for the Am positions: 50 (one string above), 42(the same string) and 32 (one below). The similarity between positions is given by Equations 1 and 2.

$$(1)\ fSimP(PosA, PosB) = 1 - \frac{1}{e} \times fDis(PosA, PosB),$$
$$\forall PosA, PosB : PosA(string) = PosB(string)$$

$$(2)\ fSimP(PosA, PosB) = \frac{1}{2} - \frac{1}{e} \times fDis(PosA, PosB),$$
$$\forall PosA, PosB : PosA(string) \neq PosB(string)$$

Once all the similarities factors have been calculated, the chord similarity is the average of the larger values of each row of the matrix, provided that there is no intersection of the larger values in the columns. Otherwise we will choose the larger value of the conflicted factors and get the second larger value of the row that has the lower value. In the example the larger values are: 0,31 + 0,62 + 62 = 0,52.

The most similar chord shape is processed to fill the empty spaces with repeated notes, creating new chord shapes. If one of these new chord's shapes satisfies the rhythm and user constraints, then it is returned. Otherwise the next most similar chord shape is processed, and so on. The complete chord shape for G following the Am (50-42-32-21-10) is 63; 40; 30; 20; 13; which has the same polyphony.

Rhythm Pattern

Style must be quantifiable or it cannot be transcribed to code (Cope 1991). In the context of this work, the musical style concept will be simplified to a set of guitar rhythm patterns and harmonic rules. The musical style will guide the system in the chord's shape choice.

Guitar Rhythm Patterns (GRP) are usually restricted to right-hand techniques, although they can have a strong influence in the chord's shapes choice (left-hand). This bi-lateral relation is often ignored by the Instrument Performance Systems. The GRP is a set of repetitive actions, which guitarists perform when they are playing. These actions are related to the way in which they pick the strings, the direction, intensity, and the sequence. In popular music, the rhythm patterns have a very straight relation with the music style.

Two types are more common: arpeggios and strums. Basically the system differs one from the other based on the Musical Events (ME) that composes the GRP. If the number of ME's starting in a same beat is greater than the number of right-hand fingers then we have a strum. A strum is characterized by fast movements of right-hand fingers or pluck in one direction (up or down) meaning that it is not possible to jump a particular string, so the strings of the chord shape must be contiguous. Figure 2 shows an example of strum GRP, with default polyphony of 4 voices.

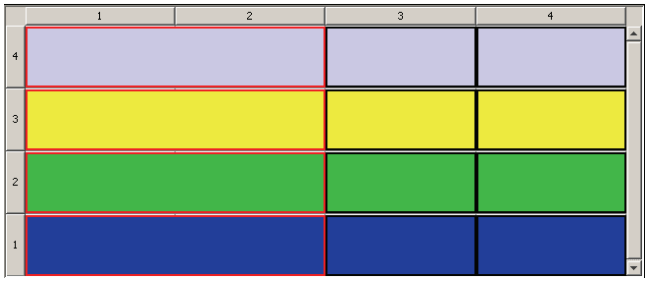

Figure 2: Strum GPR.

When the system recognizes strums GRP it can search for chord shapes with different polyphony specified by the users (between a max and a min). This is because when you are strumming the guitar you can play the same rhythm pattern starting from the 6th string or the 4th since the strings are contiguous.

Arpeggios are simpler to deal with. The number of simultaneous notes in the same beat does not exceed the number of right hand fingers meaning that we can pick the string once at time or simultaneously. The strings are not necessarily contiguous and the chord polyphony must be inferred or informed by the user.

The GPR parameters are:

Arpeggio direction: Upper or down arpeggio. Set the system to start playing the notes from high to low pitch or the opposite.

Arpeggio rate: How slow the system will play the arpeggio, this means the time gap between each note of the arpeggio. It varies based on the current time-figure note.

Swing: Time variation to more or less than 50% of the time-figure note. The goal of this parameter is to humanize the sound, entering some little delays or precipitations in some musical events.

Volume: Notes amplitude. Used to accent the beats.

Pitch(block or notes): Set to specify whether a note should be in a high, medium or low region, whenever is possible satisfy this constrain.

Polyphony: Number of simultaneous notes that the chord shape should have (resolution). A maximum and a minimum polyphony could also be defined in order to fit the chord's shapes into the GRP.

Beats: The duration of the rhythm pattern is given by the number of beats multiplied by the time-figure note value.

Time-figure note: The duration of each beat. The default is 1; the higher it is the slower is the execution.

Number of fingers: number of right hand fingers available for the execution of the rhythm pattern. If the number of rhythm pattern simultaneous note's attacks in a certain moment is larger than the number of available fingers to pull the strings then the strings must be played in a unique and fast movement. In this case, the chord shape's strings should be contiguous.

Arpeggio ignore: When an arpeggio is identified by the system, the swing parameter is disabled since the attack time will be calculated based on the arpeggio delay and it does not consider the swing value. We should use this option if we want to use the swing.

Strings blocks: For some GRP, it could be interesting to group the ME's in contiguous strings. For example: we could group the 3 higher notes in contiguous strings and let the bass note free. This GPR could be reggae for instance.

RESULTS

The system was implemented in Java J2SE 1.5 using JavaSound and jMusic v1.4.2 to deal with MIDI. During the implementation a number of music students,

professional and non-professionals musicians helped us evaluate the system as beta testers. (Some MIDI files generated by the system are available at http://cmr.soc.plymouth.ac.uk/llcostalonga/music.)
The evaluation was restricted to the usability of the prototype to generate music material for further compositions. Thirty users were trained in two separated groups: guitar players and non-guitar players. Questionnaire and interviews were used for feed-back. The main problem reported by the testers was related to the difficulty in playing a well-known musical piece only by setting its chords and rhythm pattern.
Non-guitarists found it more useful and reliable than the guitarists. 46% of the non-guitarists said that the system can play like a human, but only 26% of the guitarist said that they could use the same chord shapes. However, 76% of the total group said the system played idiomatically. This is the most important measure for us because we are extending the limits of human performance finding new ways of playing considering their constrains.
As an example, we submit the harmony of "Girl from Ipanema" (Vinicius de Moraes e Tom Jobim) with a Bossa Nova GRP and the result was G7M(9) 63; 50; 44; 34; A7 65; 57; 45; 36; Am7 65; 57; 45; 35; D7(b9) 55; 44; 35; 24; G7M 63; 44; 34; 23; D7(add13) 40; 34; 21; 12;
Although the chord's shapes are correct, they are not the mostly common used. We believe that the results will be improved in the next stage of the research when the system starts to learn from performance information from real examples played by humans.

Conclusion and Future work

In this paper we have presented a system that is capable of identifying chords textual notation and generate fingering for a particular string-fretted instrument considering the musician's restrictions and also the musical style in terms of the rhythm pattern. Currently, we are improving the system with the ability to consider expressiveness parameters of a specific guitar player in a particular musical style to generate music that sounds more "human".

We believe that it is possible to change the rendering of expressive intentions of a musical performance by computing suitable deviations from a neutral performance (Hörnel and Menzel, 1998). We are currently adding expression parameters to the system. The system will learn from real guitar performances aiming to find the differences between two (or more) performances of the same piece, but played by different interpreters. With this knowledge we believe that the system will be able to mimic a particular guitarist in a particular musical style.

This research is sponsored by CAPES – Ministry of Education of Brazil.

References

Burton, A. R., and Vladimirova, T. Genetic Algorithm Utilizing Neural Network Fitness Evaluation for Musical Compositions. In *Proceedings of International Conference on Artificial Neural Networks and Genetic Algorithms*, 1997, pp. 220—224

Chediak, A. 1984. Dicionário de Acordes Cifrados*: Harmonia Aplicada a Música Popular*. Irmãos Vitale.(in portuguese)

Cope, D. 1991. *Computer and Musical Style*. Oxford Press.,.

Gabrielsoon, A. 1997. *Music Performance*. The Psychology of Music. In. D. Deutsh, ed. The Psychology of Music, 2nd ed. New York: Academic Press.

Goebl, W., Pampalk, E., and Widmer, G. 2004. Exploring expressive performance strajectories: Six famous pianists play six Chopin pieces. In Proceedings of the 8th International Conference on Music Perception and Cognition Evanston, IL, CD-ROM, Causal Productions, Adelaide, pp. 505--509.

Hörnel, D., Menzel, W. 1998. Learning Musical Structure and Style with Neural Network. *Computer Music Journal* 22(4), pp. 44-62.

jMusic- *Computer music* composition *in Java* . Available at http://jmusic.ci.qut.edu.au/. Accessed on Nov 2005.

Meredith, D., Wiggins, G., Lemtröm, K. 2001. Pattern induction and matching in polyphonic music and other multidimensional datasets, In *Proceedings of the Fifty World Multiconference on Systemics Cybernetics and Informatics* (SCI2001), v.10, Orlando, FL, ,pp. 61-66.

Miranda, E. R. 1993. Cellular Automata Music: An Interdisciplinary Project, *Interface*, Vol. 22, No. 1, pp. 3-21.

Repp, B.H. 1990. Pattern of Expressive Timing in Performances of a Beethoven Minuet by Nineteen Famous Pianists. *Journal of the Acoustical Society of America* 88, , pp. 622-641.

Rolland, P. 2001. FlExPat: Flexoble Extraction of Sequencial Patterns, In *Proceedings of the IEEE International Conference on Data* Mining, Washington, DC, USA Pages: 481 – 488.

Widmer, G. 2003. Discovering simple rules in complex data: A meta-learning algorithm and some surprising music discoveries. *Artificial Intelligence* 146, pp. 129-148.

Zanon, P., DePoli, G. 2003 Estimation of Parameters in Rule Systems for Expressive Rendering of Musical Performance. *Computer Music Journal* 27(1), pp. 29-46.