Inducing a Generative Expressive Performance Model using a Sequential-Covering Genetic Algorithm

Rafael Ramirez
Pompeu Fabra University
Music Technology Group
Ocata 1, 08003 Barcelona
Spain
rafael@iua.upf.edu

Amaury Hazan
Pompeu Fabra University
Music Technology Group
Ocata 1, 08003 Barcelona
Spain
hazan@iua.upf.edu

ABSTRACT

In this paper, we describe an evolutionary approach to inducing a generative model of expressive music performance for Jazz saxophone. We begin with a collection of audio recordings of real Jazz saxophone performances from which we extract a symbolic representation of the musician's expressive performance. We then apply an evolutionary algorithm to the symbolic representation in order to obtain computational models for different aspects of expressive performance. Finally, we use these models to automatically synthesize performances with the expressiveness that characterizes the music generated by a professional saxophonist.

Categories and Subject Descriptors

J.5 [**Computer Applications**]: Arts and Humanities

General Terms

Algorithms

Keywords

Genetic Algorithms, Expressive Music Performance

1. INTRODUCTION

Evolutionary computation [6] is being considered with growing interest in musical applications. One of the music domains in which evolutionary computation has made most impact is music composition. A large number of evolutionary systems for composing musical material have been proposed (e.g. [3, 34]). In addition to music composition, evolutionary computing has been considered in music improvisation applications where an evolutionary algorithm typically models a musician improvising (e.g. [2]). Nevertheless, little research focusing on the use of evolutionary computation for expressive performance analysis has been reported (e.g. [9]).

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. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
GECCO'07 July 7–11, 2007, London, England, United Kingdom.
Copyright 2007 ACM 978-1-59593-697-4/07/0007 ...$5.00.

In the past, the main approaches to study expressive performance have been empirical approaches based on statistical analysis (e.g. [31]), mathematical modeling (e.g. [33]), and analysis-by-synthesis (e.g. [8]). In all these approaches, it is a person who is responsible for devising a theory or a mathematical model which captures different aspects of musical expressive performance. The theory or model is later tested on real performance data in order to determine its accuracy.

In this paper we describe an approach to investigate musical expressive performance based on evolutionary computation. Instead of manually modeling expressive performance and testing the model on real musical data, we let a computer use a sequential covering genetic algorithm to automatically induce expressive performance models from real performance data: audio recordings of Jazz standards. The algorithm incrementally constructs a set of rules by learning new rules one at a time, removing the positive examples covered by the latest rule before attempting to learn the next rule. The algorithm provides an interpretable specification of the expressive principles applied to a music interpretation and, at the same time, it provides a generative models of expressive performance, i.e. a model capable of endowing a computer generated music performance with the timing and energy expressiveness that characterizes the music generated by a professional saxophonist.

The use of evolutionary techniques for modeling expressive music performance provides a number of potential advantages over other supervised learning algorithms. By applying our evolutionary algorithm, it is possible to (1) explore and analyze the induced expressive model as it 'evolves', (2) guide and interact with the evolution of the model, and (3) obtain different models resulting from different executions of the algorithm. This last point is very relevant to the task of modeling expressive music performance since it is desirable to obtain a non-deterministic model capturing the different possible interpretations a performer may produce for a given piece.

The rest of the paper is organized as follows: Section 2 describes how we extract a set of acoustic features from the audio recordings in order to obtain a symbolic description of the different expressive parameters embedded in the recordings. In Section 3, we describe our evolutionary approach for inducing expressive music performance computational models for different aspects of Expressive performance. Section 4 reports on related work, and finally Section 5 presents some conclusions and indicates some areas of future research.

2. MELODIC DESCRIPTION

In this section, we summarize how we extract a symbolic description from the monophonic recordings of performances of Jazz standards. We need this symbolic representation in order to apply a sequential covering genetic algorithm to the data. In this paper, our interest is to model note-level transformations such as onset deviations, duration transformations and energy variations. Thus, descriptors providing note-level information are of particular interest.

2.1 Algorithms for feature extraction

First of all, we perform a spectral analysis of a portion of sound, called an analysis frame, whose size is a parameter of the algorithm. This spectral analysis lies in multiplying the audio frame with an appropriate analysis window and performing a Discrete Fourier Transform (DFT) to obtain its spectrum. In this case, we use a frame width of 46 ms, an overlap factor of 50%, and a Keiser-Bessel 25dB window. Then, we perform a note segmentation using low-level descriptor values. Once the note boundaries are known, the note descriptors are computed from the low-level and the fundamental frequency values.

2.2 Low-level descriptors computation

The main low-level descriptors used to characterize expressive performance are instantaneous energy and fundamental frequency.

The energy descriptor is computed on the spectral domain, using the values of the amplitude spectrum at each analysis frame. In addition, energy is computed in different frequency bands as defined in [13], and these values are used by the algorithm for note segmentation.

For the estimation of the instantaneous fundamental frequency we use a harmonic matching model derived from the Two-Way Mismatch procedure (TWM) [17]. For each fundamental frequency candidate, mismatches between the harmonics generated and the measured partials frequencies are averaged over a fixed subset of the available partials. A weighting scheme is used to make the procedure robust to the presence of noise or absence of certain partials in the spectral data. The solution presented in [17] employs two mismatch error calculations. The first one is based on the frequency difference between each partial in the measured sequence and its nearest neighbor in the predicted sequence. The second is based on the mismatch between each harmonic in the predicted sequence and its nearest partial neighbor in the measured sequence. This two-way mismatch helps to avoid octave errors by applying a penalty for partials that are present in the measured data but are not predicted, and also for partials whose presence is predicted but which do not actually appear in the measured sequence. The TWM mismatch procedure has also the benefit that the effect of any spurious components or partial missing from the measurement can be counteracted by the presence of uncorrupted partials in the same frame.

First, we perform a spectral analysis of all the windowed frames, as explained above. Secondly, the prominent spectral peaks of the spectrum are detected from the spectrum magnitude. These spectral peaks of the spectrum are defined as the local maxima of the spectrum which magnitude is greater than a threshold. The spectral peaks are compared to a harmonic series and a two-way mismatch (TWM) error is computed for each fundamental frequency candidates. The candidate with the minimum error is chosen to be the fundamental frequency estimate.

After a first test of this implementation, some improvements to the original algorithm where implemented to deal with some errors of the algorithm:

- Peak selection: a peak selection routine has been added in order to eliminate spectral peaks corresponding to noise. The peak selection is done according to a masking threshold around each of the maximum magnitude peaks. The form of the masking threshold depends on the peak amplitude, and uses three different slopes depending on the frequency distance to the peak frequency.
- Context awareness: we take into account previous values of the fundamental frequency estimation and instrument dependencies to obtain a more adapted result.
- Noise gate: a noise gate based on some low-level signal descriptor is applied to detect silences, so that the estimation is only performed in non-silent segments of the sound.

2.3 Note segmentation

Note segmentation is performed using a set of frame descriptors, which are energy computations in different frequency bands and fundamental frequency. Energy onsets are first detected following a band-wise algorithm that uses some psycho-acoustical knowledge [13]. In a second step, fundamental frequency transitions are also detected. Finally, both results are merged to find the note boundaries (onset and offset information).

2.4 Note descriptor computation

We compute note descriptors using the note boundaries and the low-level descriptors values. The low-level descriptors associated to a note segment are computed by averaging the frame values within this note segment. Pitch histograms have been used to compute the pitch note and the fundamental frequency that represents each note segment, as found in [18]. This is done to avoid taking into account mistaken frames in the fundamental frequency mean computation.

2.5 Intra-note segment characterization

Once we have found the intra-note segment limits, we describe each one by its duration (absolute and relative to note duration), start and end times, initial and final energy values (absolute and relative to note maximum) and slope. For the stable part of each note (sustain segment), we extract an averaged spectral centroid and spectral tilt in order to have timbral descriptors related to the brightness of a particular execution. We compute the spectral centroid as the frequency bin corresponding to the barycenter of the spectrum, expressed as (6), where fft is the fast fourier transform of a frame, N is the size of the fast fourier tarnsform, and k is the bin index. For the spectral tilt, we perform a linear regression of the logarithmic spectral envelope between 2kHz and 6kHz, and get the slope expressed in dB/Hz.

$$SC = \frac{\sum_{k=1}^{N} k|fft(k)|}{\sum_{k=1}^{N} |fft(k)|} \tag{1}$$

2.6 Note clustering

Once each of the notes in the audio recordings has been characterized by its intra-note features as described above, we proceed to apply a k-means clustering algorithm to identify groups of similar notes. This clustering of notes is motivated by the fact that we are interested in devising a mechanism to determine in which musical context a particular type of note (e.g. a note with a very sharp attack) should be played. In the folowing section we tackle this problem by inducing a classifier whose input is a particular note musical context and its output is a class representing a particular cluster of notes. For synthesis purposes we simply select the most convenient note within the selected cluster and modify the note to fit our context.

3. LEARNING THE EXPRESSIVE PERFORMANCE MODEL

In this section, we describe our inductive approach for learning an expressive music performance model from performances of Jazz standards. Our aim is to obtain a model capable of endowing a computer generated music performance with the expressiveness that characterizes human generated music. That is to say, we intend to generate automatically human-like expressive performances of a piece given an inexpressive description of the piece (e.g. a textual description of its score). We are aware of the fact that not all the expressive transformations performed by a musician can be predicted at a local note level. Musicians perform music considering a number of abstract structures (e.g. musical phrases) which makes of expressive performance a multi-level phenomenon. In this context, our aim is to obtain a computational model of expressive performance which combines note-level and structure-level information. As a first step in this direction, we have based our musical analysis on the implication/realization model, proposed by Narmour [26]. The Implication/Realization model is a theory of perception and cognition of melodies. The theory states that a melodic musical line continuously causes listeners to generate expectations of how the melody should continue. Any two consecutively perceived notes constitute a melodic interval and if this interval is not conceived as complete, it is an *implicative interval*, i.e. an interval that implies a subsequent interval with certain characteristics. That is to say, some notes are more likely than others to follow the implicative interval. Two main principles recognized by Narmour concern *registral direction* and *intervallic difference*. The principle of registral direction states that small intervals imply an interval in the same registral direction (a small upward interval implies another upward interval and analogously for downward intervals), and large intervals imply a change in registral direction (a large upward interval implies a downward interval and analogously for downward intervals). Based on these two principles, melodic patterns or groups can be identified that either satisfy or violate the implication as predicted by the principles. Figure 1 shows prototypical Narmour structures.

A note in a melody often belongs to more than one structure, i.e. a description of a melody as a sequence of Narmour structures consists of a list of overlapping structures. We parse each melody in the training data in order to automatically generate an implication/realization analysis.Figure 2 shows the analysis for a fragment of *All of me*.

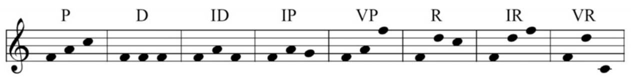

Figure 1: Prototypical Narmour structures

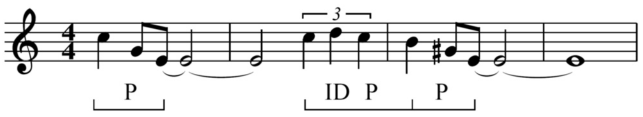

Figure 2: Narmour analysis of *All of Me*

3.1 Training data

The training data used in our experimental investigations are monophonic recordings of four Jazz standards (Body and Soul, Once I Loved, Like Someone in Love and Up Jumped Spring) performed by a professional musician at 11 different tempos around the nominal tempo. For each piece, the nominal tempo was determined by the musician as the most natural and comfortable tempo to interpret the piece. Also for each piece, the musician identified the fastest and slowest tempos at which a piece could be reasonably interpreted. Interpretations were recorded at regular intervals around the nominal tempo (5 faster and 5 slower) within the fastest-slowest tempo limits. The data set is composed of 4360 performed notes. Each note in the training data is annotated with its corresponding performed characteristics and a number of attributes representing both properties of the note itself and some aspects of the context in which the note appears. Information about the note include note duration and the note metrical position within a bar, while information about its melodic context include performed tempo, information on neighboring notes as well as the Narmour group in which the note appears in third position.

3.2 Learning task

In this paper, we are concerned with note-level expressive transformations, in particular transformations of note duration, onset, energy and note type. Initially, for each expressive transformation, we approach the problem as a classification problem, e.g. for note duration transformation we classify each note to belong to one of the classes lengthen, shorten or same. Once we obtain a classification mechanism capable of classifying all notes in our training data, we apply a regression algorithm in order to produce a numerical value representing the amount of transformation to be applied to a particular note (in the case of note type we apply a nearest neighbor algorithm to obtain the index of the most suitable note within the selected cluster). The complete algorithm is detailed in the next section.

The performance classes that interest us are *lengthen, shorten* and *same* for duration transformation, *advance, delay* and *same* for onset deviation, and *soft, loud* and *same* for energy variation. A note is considered to belong to class lengthen, if its performed duration is 20% longer (or more) that its nominal duration, e.g. its duration according to the score. Class shorten is defined analogously. A note is considered to be in class advance if its performed onset is 5% of a bar earlier (or more) than its nominal onset. Class delay is defined analogously. A note is considered to be in class loud if it is played louder than its predecessor and louder than the average level of the piece. Class soft is defined analogously. We decided to set these boundaries after experimenting with different ratios. The main idea was to guarantee that a note classified, for instance, as lengthen was purposely lengthened by the performer and not the result of a performance inexactitude. In the case of note type prediction, the classes of interest are the different *clusters* containing individual segmented notes.

3.3 Algorithm

We applied a genetic sequential covering algorithm to the training data. Roughly, the algorithm incrementally constructs a set of rules by learning new rules one at a time, removing the positive examples covered by the latest rule before attempting to learn the next rule. Rules are learned using a genetic algorithm with the usual parameters r, m and p respectively determining the fraction of the parent population replaced by crossover, the mutation rate, and population size. We set these parameters as follows: $r = 0.8$, $m = 0.05$ and $p = 200$. For each class of interest (e.g. *lengthen*, *shorten*, *same*), we collect the rules with best fitness during the evolution of the population. For obtaining rules for a particular class of interest (e.g. lengthen) we consider s negative examples the examples of the other two complementary classes (e.g. shorten and same). It is worth mentioning that although the test was running over 40 generations, the fittest rules were obtained around the 20th generation. Once we obtain the set of rules covering all the training examples, for each rule, we apply linear regression to the examples covered by the rule in order to obtain a linear equation predicting a numerical value. This leads to a set of rules producing a numerical prediction and not just a nominal class prediction. In the case of note type prediction, we apply 1-nearest neighbor within the predicted cluster in order to obtain the index of the most 'suitable' note in the cluster. Since we are interested in selecting a note for synthesis purposes, the most 'suitable' note in a particular cluster is the note that requires least transformation (i.e. pitch transposing and time stretching) to satisfy the pitch, duration, onset and energy requirements. The algorithm is as follows:

```
GeneticSeqCovAlg(Class,Fitness,Threshold,
                            p,r,m,Examples)
Pos = examples which belong to Class
Neg = examples which do not belong to Class
Learned_rules = {}
While Pos do
   P = generate p hypothesis at random
   for each h in P, compute fitness(h)
   while max(fitness(h)) < Threshold do
      create a new generation Pnew
      P = Pnew
      for each h in P, compute fitness(h)
   Return the hypothesis Newrule from P
            that has the highest fitness
   Rpos = members of Pos covered by NewRule
   NumericNewRule = NewRule with Class
                replaced by NumericValue
   Learned_rules = Learned_rules
                    + NumericNewrule
   Pos = Pos - Rpos
Return Learned_rules
```

The outer loop learns new rules one at a time, removing the positive examples covered by the latest rule before attempting to learn the next rule. The inner loop performs a genetic search through the space of possible rules in search of a rule with high accuracy. At each iteration, the outer loop adds a new rule to its disjunctive hypothesis, `Learned_rules`. The effect of each new rule is to generalize the current disjunctive hypothesis (i.e. increasing the number of instances it classifies as positive) by adding a new disjunct. At this level, the search is a specific-to-general search starting with the most specific hypothesis (i.e. the empty disjunction) and terminating when the hypothesis is sufficiently general to cover all training examples. In the case of duration, onset and energy rules, `NumericNewRule` is a rule where the consequent `NumericValue` is a linear equation $X = w_0 + w_1 * a_1 + w_2 * a_2 + \cdots + w_k * a_k$ where X is the predicted value expressed as a linear combination of the attributes $a_1, \ldots, a_k$ of the training examples with predetermined weights $w_0, \ldots, w_k$. The weights are calculated using the set of positive examples covered by the rule `Rpos` by linear regression. In the case of note type rules, `NumericValue` is the index of a note in the predicted cluster determined by the nearest neighbor algorithm using the distance

$$\sqrt{PitchDif^2 + DurationDif^2}$$

where $PitchDif$ is the normalized difference between the pitch of the score note and the pitch of a note in the predicted cluster. Similarly $DurationDif$ is the normalized difference between the duration of the score note and the duration of a note in the predicted cluster. The rationale for this distance measure is that for audio synthesis quality purposes we are interested in minimizing the processing of the samples in the clusters.

The inner loop performs a fine grained search to determine the exact form of each new rule. In the inner loop, a new generation is created as follows:

- Select: probabilistically select $(1 - r)p$ members of `P` to add to `Ps`. The probability of $Pr(h_i)$ of selecting hypothesis h_i from `P` is

$$Pr(h_i) = \frac{Fitness(h_i)}{\Sigma(h_j)} \qquad (1 \leq j \leq p)$$

- Crossover: probabilistically select $(r * p)/2$ pairs of hypothesis from `P` (according to $Pr(h_i)$ above). For each pair, produce an offspring by applying the crossover operator (see below) and add it to `Ps`.

- Mutate: Choose m percent of the members of `Ps` with uniform probability and apply the mutation operator (see below).

Hypothesis representation. The hypothesis space of rule preconditions consists of a conjunction of a fixed set of attributes. Each rule is represented as a bit-string as follows: the previous and next note duration are represented each by five bits (i.e. much shorter, shorter, same, longer and much longer), previous and next note pitch are represented each by five bits (i.e. much lower, lower, same, higher and much higher), metrical strength by five bits (i.e. very weak, weak, medium, strong and very strong), tempo by three bits (i.e. slow, nominal and fast) and Narmour group by three bits. For example in our representation the rule *"if the previous note duration is much longer and its pitch is the same and it is in a very strong metrical position and the current note appears in Narmour group R then lengthen the duration of the current note"*

is coded as the binary string:

00001 11111 00100 11111 00001 111 110 001

The exact meaning of the adjectives which the particular bits represent are as follows: previous and next note durations are considered much shorter if the duration is less than half of the current note, shorter if it is shorter than the current note but longer than its half, and same if the duration is the same as the current note. Much longer and longer are defined analogously. Previous and next note pitches are considered much lower if the pitch is lower by a minor third or more, lower if the pitch is within a minor third, and same if it has same pitch. Higher and much higher are defined analogously. The note's metrical position is very strong, strong, medium, weak, and very weak if it is on the first beat of the bar, on the third beat of the bar, on the second or fourth beat, offbeat, and in none of the previous, respectively. The piece was played at slow, nominal, and fast tempos if it was performed at a speed slower of more than 15% of the nominal tempo (i.e. the tempo identified as the most natural by the performer), within 15% of the nominal tempo, and faster than 15% of the nominal tempo, respectively. In the case of the note's Narmour groups we decided to code only one Narmour group for each note. This is, instead of specifying all the possible Narmour groups for a note, we select the one in which the note appears in third position (if there is no such group, we consider one in which the note appears either in first or second position, in that order).

Genetic operators. We use standard single-point crossover and mutation operators with two restrictions. In order to perform a crossover operation of two parents the crossover points are chosen at random as long as they are on the attributes sub string boundaries. Similarly the mutation points are chosen randomly as long as they do not generate inconsistent rule strings, e.g. only one class can be predicted so exactly one 1 can appear in the last three bit sub string.

Fitness function. The fitness of each hypothesized rule is based on its classification accuracy over the training data. In particular, the function used to measure fitness is

$$\frac{tp^{\alpha}}{(tp+fp)}$$

where tp is the number of true positives, fp is the number of false positives, and α is a constant which controls the true positives to false positives ratio. Often α is set to 1 which results in the standard fitness function

$$\frac{tp}{(tp+fp)}$$

This fitness function favors individuals covering a small number of true positive and 0 false positives (resulting in a fitness value of 1) over individuals covering a large number of true positives and 1 false positive (resulting in a fitness value of less than 1). In our application this is an undesirable property of the fitness function since we are interested in inducing general expressive performance rules covering a large number of examples (possibly including a small number of false positives). Thus, in our algorithm we have set $\alpha = 1.15$ which, for our application, is a good compromise between coverage and accuracy.

3.4 Results

It is always difficult to evaluate formally a model which captures subjective knowledge, as it is the case of an expressive music performance model. The ultimate evaluation may consist of listening to the transformations the model performs. Alternatively, the model may be evaluated by comparing the model's transformation predictions and the actual transformations performed by the musician. Figure 3 shows the note-by-note duration ratio predicted a model induced by executing the algorithm and compare it with the actual duration ratio in the recording. Similar results were obtained for the predicted onset deviation and energy variation. As illustrated by Figure 3 the induced model seems to accurately capture the musician's expressive performance transformations (despite the relatively small amount of training data). The correlation coefficient for the onset, duration and energy sub models is 0.75, 0.84 and 0.86, respectively. These numbers were obtained by performing a 10-fold cross validation on the data. At each fold, we removed the performances similar to the ones selected in the test set, i.e. the performances of the same piece at similar tempos. We ran the sequential covering genetic algorithm 20 times in order to observe the differences between the correlation coefficient of different runs. We observed no substantial differences.

The use of an evolutionary algorithm for inducing an expressive performance model provides the possibility of examining the model as it 'evolves'. This is, it is possible to examine and interpret the rules induced by the algorithm at each population generation. In particular, we are interested in interpreting high-accuracy rules in late populations (25th generation or later). The evolutionary approach to expressive performance modeling permits to guide the model search in a natural way. For instance, we have experimented by imposing restrictions on the general shape of rules in order to prioritize simple interpretable rules, e.g. rules with blocks of 1's.

We examined some of the classification rules the algorithm induced (before replacing the class with the numerical predicted value). Some of these rules proved to be of musical interest and correspond to intuitive musical knowledge. In order to illustrate the types of rules found let us consider an example duration rule:

RULE: 11111 01110 11110 00110 00011 010 010 001

"At nominal tempo, if the duration of the next note is similar and the note is in a strong metrical position and the note appears in a D Narmour group then lengthen the current note."

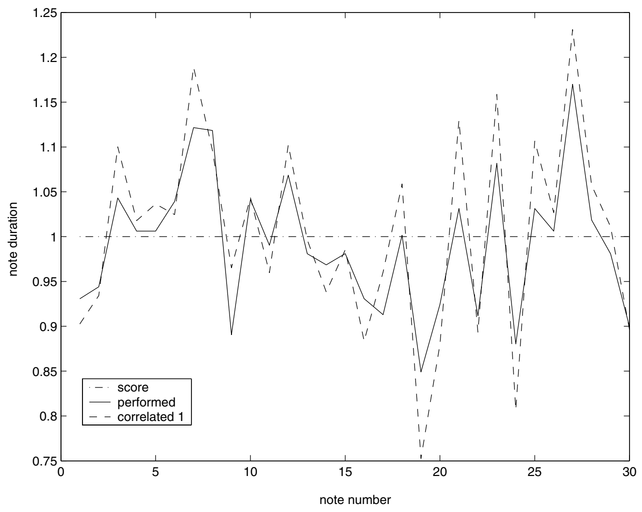

Figure 3: Correlation between model predicted duration values and the actual performed values in *Body and Soul* at a tempo of 65

4. RELATED WORK

Evolutionary computation has been considered with growing interest in musical applications [25]. A large number of experimental systems using evolutionary techniques to generate musical compositions have been proposed, including Cellular Automata Music [23], a Cellular Automata Music Workstation [12], CAMUS [24], MOE [5], GenDash [35], CAMUS 3D [21], Vox Populi [20], Synthetic Harmonies [2], Living Melodies [4] and Genophone [19]. Composition systems based on genetic algorithms generally follow the standard genetic algorithm approach for evolving musical materials such as melodies, rhythms and chords. Thus, such composition systems share the core approach with the one presented in this chapter. For example, Vox Populi [20] evolves populations of chords of four notes, each of which is represented as a 7-bit string. The genotype of a chord therefore consists of a string of 28 bits (e.g. 1001011 0010011 0010110 0010101) and the genetic operations of crossover and mutation are applied to these strings in order to produce new generations of the population. The fitness function is based on three criteria: melodic fitness, harmonic fitness and voice range fitness. The melodic fitness is evaluated by comparing the notes of the chord to a reference value provided by the user, the harmonic fitness takes into account the consonance of the chord, and the voice range fitness measures whether or not the notes of the chord are within a range also specified by the user. Evolutionary computation has also been considered for improvisation applications such as [2], where a genetic algorithm-based model of a novice Jazz musician learning to improvise was developed. The system evolves a set of melodic ideas that are mapped into notes considering the chord progression being played. The fitness function can be altered by the feedback of the human playing with the system.

Nevertheless, very few works focusing on the use of evolutionary computation for expressive performance analysis have been done. In the context of the ProMusic project, Grachten et al. [9] optimized the weights of edit distance operations by a genetic algorithm in order to annotate a human Jazz performance. They present an enhancement of edit distance based music performance annotation. In order to reduce the number of errors in automatic performance annotation, they use an evolutionary approach to optimize the parameter values of cost functions of the edit distance. In another study, Hazan et al. [10] proposed an evolutionary generative regression tree model for expressive rendering MIDI performances. Madsen at al. [16] present an approach to exploring similarities in music classical piano performances based on simple measurements of timing and intensity in 12 recordings of a Schubert piano piece. The work presented in this chapter is an extension to our previous work [30] where we induce expressive performance classification rules using a genetic algorithm. Here, in addition to considering classification rules, we consider regression rules, and while in [30] rules are independently induced by the genetic algorithm, here we apply a sequential covering algorithm in order to cover the whole example space.

There have been several approaches for addressing expressive music performance using machine learning techniques other than evolutionary techniques. The most related work to the research presented in this chapter is the work by Ramirez et al. [28, 29] and Lopez de Mantaras et al. [14].

Lopez de Mantaras et al. report on SaxEx, a performance system capable of generating expressive solo performances in jazz. Their system is based on case-based reasoning, a type of analogical reasoning where problems are solved by reusing the solutions of similar, previously solved problems. In order to generate expressive solo performances, the case-based reasoning system retrieves, from a memory containing expressive interpretations, those notes that are *similar* to the input inexpressive notes. The case memory contains information about metrical strength, note duration, and so on, and uses this information to retrieve the appropriate notes. However, their system does not allow one to examine or understand the way it makes predictions.

Ramirez et al. explore and compare different machine learning techniques for inducing both, an *interpretable* expressive performance model (characterized by a set of rules) and a *generative* expressive performance model. Based on this, they describe a performance system capable of generating expressive monophonic Jazz performances and providing 'explanations' of the expressive transformations it performs. The work described in this chapter has similar objectives but by using a genetic algorithm it incorporates some desirable properties: (1) the induced model may be explored and analyzed while it is 'evolving', (2) it is possible to guide the evolution of the model in a natural way, and (3) by repeatedly executing the algorithm different models are obtained. In the context of expressive music performance modeling, these properties are very relevant.

With the exception of the work by Lopez de Mantaras et al. and Ramirez et al., most of the research in expressive performance using machine learning techniques has focused on classical piano music where the tempo of the performed pieces is not constant and melody alterations are not permitted (in classical music melody alterations are considered performance errors). Thus, in these works the focus is on global tempo and energy transformations while we are interested in note-level tempo and energy transformations as well as in melody ornamentations which are a very important expressive resource in Jazz.

Widmer [36] reported on the task of discovering general rules of expressive classical piano performance from real performance data via inductive machine learning. The performance data used for the study are MIDI recordings of 13 piano sonatas by W.A. Mozart performed by a skilled pianist. In addition to these data, the music score was also coded. The resulting substantial data consists of information about the nominal note onsets, duration, metrical information and annotations. When trained on the data an inductive rule learning algorithm discovered a small set of quite simple classification rules that predict a large number of the note-level choices of the pianist.

Tobudic et al. [32] describe a relational instance-based approach to the problem of learning to apply expressive tempo and dynamics variations to a piece of classical music, at different levels of the phrase hierarchy. The different phrases of a piece and the relations among them are represented in first-order logic. The description of the musical scores through predicates (e.g. *contains(ph1,ph2)*) provides the background knowledge. The training examples are encoded by another predicate whose arguments encode information about the way the phrase was played by the musician. Their learning algorithm recognizes similar phrases from the training set and applies their expressive patterns to a new piece.

Other inductive machine learning approaches to rule learning in music and musical analysis include [7] and [1]. In [7], Dovey analyzes piano performances of Rachmaniloff pieces using inductive logic programming and extracts rules underlying them. In [1], Van Baelen extended Dovey's work and attempted to discover regularities that could be used to generate MIDI information derived from the musical analysis of the piece.

5. CONCLUSION

This paper describes an evolutionary computation approach for learning an expressive performance model from recordings of Jazz standards by a skilled saxophone player. Our objective has been to find a computational model which predict how a particular note in a particular context should be played (e.g. longer or shorter than its nominal duration). In order to induce the expressive performance model, we have extracted a set of acoustic features from the recordings resulting in a symbolic representation of the performed pieces and then applied a sequential-covering genetic algorithm to the symbolic data and information about the context in which the data appear. In addition, some of the classification rules induced by the algorithm proved to be of musical interest. We plan to increase the amount of training data as well as experiment with different information encoded in it. Increasing the training data, extending the information in it and combining it with background musical knowledge will certainly generate a more accurate model. Another research line is to extend our model to be able to base its prediction on expressive features such as vibrato and tremolo. We plan to induce models by extending the notes' characterization by such expressive features.

6. ACKNOWLEDGMENTS

This work is supported by the Spanish Ministry of Science and Education project ProSeMus (TIN2006-14932-C02-01). We would like to thank Maarten Grachten, Emilia Gomez and Esteban Maestre for processing the data.

7. REFERENCES

[1] Van Baelen, E. and De Raedt, L. (1996). Analysis and Prediction of Piano Performances Using Inductive Logic Programming. International Conference in Inductive Logic Programming, 55-71.

[2] Biles, J. A. (1994). GenJam: A genetic algorithm for generating Jazz solos. In ICMC Proceedings 1994.

[3] Dahlstedt, P., and Nordahl, M. (1999). Living Melodies: Coevolution of Sonic Communication First Iteration Conference on Generative Processes in the Electronic Arts, Melbourne, Australia.

[4] Dahlstedt, P. and Nordhal, M. G. (2001). Living Melodies: Coevolution of Sonic Communication. Leonardo, 34(3):243-248.

[5] Degazio, B. (1999). La evolucion de los organismos musicales. In Miranda, E. R., editor, Musica y nuevas tecnologias: Perspectivas para el siglo XXI, pages 137-148, Barcelona, Spain.

[6] De Jong, K.A. et al. (1993). Using Genetic Algorithms for Concept Learning. Machine Learning, 13, 161-188

[7] Dovey, M.J. (1995). Analysis of Rachmaninoff's Piano Performances Using Inductive Logic Programming. European Conference on Machine Learning, Springer-Verlag.

[8] Friberg, A. (1995). A Quantitative Rule System for Musical Performance. PhD Thesis, KTH, Sweden.

[9] Grachten, M., Luis Arcos, J., and Lopez de Mantaras, R. (2004). Evolutionary Optimization of Music Performance Annotation.

[10] Amaury Hazan, Rafael Ramirez, Esteban Maestre, Alfonso Perez, Antonio Pertusa (2006). Modelling Expressive Performance: A Regression Tree Approach Based on Strongly Typed Genetic Programming. EvoWorkshops, LNCS, pp.676-687

[11] Horner, A., and Goldberg, 1991. Genetic Algorithms and Computer-Assisted Music Composition, in proceedings of the 1991 International Computer Music Conference, pp. 479-482.

[12] Hunt, A., Kirk, R. and Orton, R. (1991). Musical Applications of a Cellular Automata Workstation. In

Proceedings of the International Computer Music Conference ICMC91, pages 165-166, ICMA, San Francisco, California.

[13] Klapuri, A. (1999). Sound Onset Detection by Applying Psychoacoustic Knowledge, Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP.

[14] Lopez de Mantaras, R. and Arcos, J.L. (2002). AI and music, from composition to expressive performance, AI Magazine, 23-3.

[15] Maestre, E. Hazan, A. Ramirez, R. Perez, A. 2006. Using concatenative synthesis for expressive performance in jazz saxophone, Proceedings of International Computer Music Conference.

[16] Madsen, S. T. and Widmer, G. 2005. Exploring similarities in Music Performances with an Evolutionary Algorithm. Proceedings of The 18th International FLAIRS Conference, AAAI Press.

[17] Maher, R.C. and Beauchamp, J.W. (1994). Fundamental frequency estimation of musical signals using a two-way mismatch procedure, Journal of the Acoustic Society of America, vol. 95 pp. 2254-2263.

[18] McNab, R.J., Smith Ll. A. and Witten I.H., (1996). Signal Processing for Melody Transcription,SIG working paper, vol. 95-22.

[19] Mandelis, J. (2001). Genophone: An Evolutionary Approach to Sound Synthesis and Performance. In Bilotta, E., Miranda, E. R., Pantano, P. and Todd, P. M., editors, Proceedings of ALMMA 2002 Workshop on Artificial Models for Musical Applications, pages 108-119, Editoriale Bios, Cosenza, Italy.

[20] Manzolli, J., Moroni, A., von Zuben, F. and Gudwin, R. (1999). An Evolutionary Approach Applied to Algorithmic Composition. In Miranda, E. R. and Ramalho, G. L., editors, Proceedings of VI Brazilian Symposium on Computer Music, pages 201-210, SBC/Entre Lugar, Rio de Janeiro, Brazil.

[21] McAlpine, K., Miranda, E. R. and Hogar, S. (1999). Composing Music with Algorithms: A Case Study System. Computer Music Journal, 23(2):19-30.

[22] McNab, R.J., Smith Ll. A. and Witten I.H., (1996). Signal Processing for Melody Transcription, SIG working paper, vol. 95-22.

[23] Millen, D. (1990). Cellular Automata Music. In Proceedings of the International Computer Music Conference ICMC'90, pages 314-316, ICMA, San Francisco, California.

[24] Miranda, E. R. (1993). Cellular Automata Music: An Interdisciplinary Music Project. Interface (Journal of New Music Research), 22(1):03-21.

[25] Miranda, E. R. (2004). At the Crossroads of Evolutionary Computation and Music: Self-Programming Synthesizers, Swarm Orchestras and the Origins of Melody, Evolutionary Computation 12(2): 137-158.

[26] Narmour, E. (1990). The Analysis and Cognition of Basic Melodic Structures:The Implication Realization Model. University of Chicago Press.

[27] Phon-Amnuaisuk, S., and A. Wiggins, G. (1999?) The Four-Part Harmonisation Problem: A comparison between Genetic Algorithms and a Rule-Based System.

[28] Ramirez, R. Hazan, A. Gómez, E. Maestre, E. (2005). Understanding Expressive Transformations in Saxophone Jazz Performances, Journal of New Music Research, Vol. 34, No. 4, pp. 319-330.

[29] Rafael Ramirez, Amaury Hazan, Esteban Maestre, Xavier Serra, A Data Mining Approach to Expressive Music Performance Modeling, in Multimedia Data mining and Knowledge Discovery, Springer.

[30] Ramirez, R., Hazan, A. (2005) Understanding Expressive Music Performance Using Genetic Algorithms. EvoWorkshops, LNCS, Springer, pp.508-516

[31] Repp, B.H. (1992). Diversity and Commonality in Music Performance: an Analysis of Timing Microstructure in Schumann's 'Traumerei'. Journal of the Acoustical Society of America 104.

[32] Tobudic A., Widmer G. (2003). Relational IBL in Music with a New Structural Similarity Measure, Proceedings of the International Conference on Inductive Logic Programming, Springer Verlag.

[33] Todd, N. (1992). The Dynamics of Dynamics: a Model of Musical Expression. Journal of the Acoustical Society of America 91.

[34] Tokui, N., and Iba, H. (2000). Music Composition with Interactive Evolutionary Computation.

[35] Waschka II, R. (1999). Avoiding the Fitness Bottleneck: Using Genetic Algorithms to Compose Orchestral Music. In Proceedings of the International Computer Music Conference ICMC'99, pages 201-203, ICMA, San Francisco, California.

[36] Widmer, G. (2002). Machine Discoveries: A Few Simple, Robust Local Expression Principles. Journal of New Music Research 31(1), 37-50.