Rafael Ramirez, Amaury Hazan, Esteban Maestre, and Xavier Serra
Music Technology Group
Universitat Pompeu Fabra
Ocata 1, 08003 Barcelona, Spain
{rafael, ahazan, emaestre, xserra}@iua.upf.edu

A Genetic Rule-Based Model of Expressive Performance for Jazz Saxophone

Evolutionary computation (De Jong et al. 1993) is being considered with growing interest in musical applications. One of the music domains in which evolutionary computation has made the most impact is music composition. A number of evolutionary systems for composing musical material have been proposed (e.g., Horner and Goldberg 1991; Dahlstedt and Nordhal 2001). In addition to music composition, evolutionary computing has been considered in music improvisation applications where an evolutionary algorithm typically models a musician's improvising (e.g., Biles 1994). Nevertheless, little research focusing on the use of evolutionary computation for expressive-performance analysis has been reported.

Traditionally, expressive performance has been studied using empirical approaches based on statistical analysis (e.g., Repp 1992), mathematical modeling (e.g., Todd 1992), and analysis-by-synthesis (e.g., Friberg et al. 1998). In these approaches, humans are responsible for devising a theory or a mathematical model that captures different aspects of musical expressive performance. The theory or model is later tested on real performance data to determine its accuracy.

In this article, we describe an approach to investigating 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 execute a sequential-covering genetic algorithm to automatically discover regularities and performance principles from real performance data, consisting of audio recordings of jazz standards. The algorithm combines sequential covering (Michalski 1969) and genetic algorithms (Holland 1975). The sequential-covering component of 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 genetic component of the algorithm learns each of the new rules by applying a genetic algorithm.

The algorithm provides an interpretable specification of the expressive principles applied to an interpretation of piece of music and, at the same time, it provides a generative model of expressive performance, namely, a model capable of generating a computer-music performance with the timing and energy expressiveness that characterizes human-generated music.

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 explore and analyze the induced expressive model as it "evolves," to guide and interact with the evolution of the model, and to obtain different models resulting from different executions of the algorithm. This last point is very relevant to the task of modeling expressive music performance, because 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 this article is organized as follows. First, we report on related work and describe how we extract a set of acoustic features from the audio recordings. We then describe our evolutionary approach for inducing an expressive music-performance computational model. Finally, we present some conclusions and indicate some areas of future research.

Related Work

Evolutionary computation has been considered with growing interest in musical applications (Miranda 2004). A large number of experimental

Computer Music Journal, 32:1, pp. 38–50, Spring 2008

systems using evolutionary techniques to generate musical compositions have been proposed, including Cellular Automata Music (Millen 1990), a Cellular Automata Music Workstation (Hunt, Kirk, and Orton 1991), CAMUS (Miranda 1993), MOE (Degazio 1999), GenDash (Waschka 1999), CAMUS 3D (McAlpine, Miranda, and Hogar 1999), Vox Populi (Manzolli et al. 1999), Synthetic Harmonies (Bilotta, Pantano, and Talarico 2000), Living Melodies (Dahlstedt and Nordhal 2001), and Genophone (Mandelis 2001). Composition systems based on genetic algorithms generally follow the standard genetic-algorithm approach for evolving musical materials such as melodies, rhythms, and chords. As a result, such compositional systems share the core approach with the one presented in this article. For example, Vox Populi (Manzolli et al. 1999) evolves populations of chords of four notes, each of which is represented as a seven-bit string. The genotype of a chord therefore consists of a string of 28 bits, and the genetic operations of crossover and mutation are applied to these strings 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 the notes of the chord are within a range also specified by the user. Evolutionary computation has also been considered for improvisation applications (Biles 1994), 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, few works focusing on the use of evolutionary computation for expressive-performance analysis exist. In the context of the ProMusic project, Grachten et al. (2004) optimized the weights of edit-distance operations by a genetic algorithm to annotate a human jazz performance. They present an enhancement of edit-distance-based music-performance annotation. 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. (2006) proposed an evolutionary generative regression-tree model for expressive rendering of MIDI performances. Madsen and Widmer (2005) present an approach exploring similarities in classical piano performances based on simple measurements of timing and intensity in 12 recordings of a Schubert piano piece. The work presented in this article is an extension of our previous work (Ramirez and Hazan 2005), where we induce expressive-performance classification rules using a genetic algorithm. Here, in addition to considering classification rules, we consider regression rules, and whereas in Ramirez and Hazan, rules are independently induced by the genetic algorithm, here we apply a sequential-covering algorithm to cover the whole example space.

Other Machine-Learning Techniques

Several approaches have addressed expressive music performance using machine-learning techniques other than evolutionary techniques. The work most relevant to that presented in this article is described in Lopez de Mantaras and Arcos (2002) and Ramirez et al. (2005, 2006).

Lopez de Mantaras and Arcos (2002) describe SaxEx, a performance system capable of generating expressive solo performances of jazz. Their system is based on case-based reasoning, a type of analogical reasoning in which problems are solved by reusing the solutions of similar, previously solved problems. 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. (2007) 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 article 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. (This article is an extended version of Ramirez and Hazan 2007.)

With the exception of the work by Lopez de Mantaras and Arcos (2002) and Ramirez et al. (2005, 2006), most of the research in expressive performance using machine-learning techniques has focused on classical piano solo music (e.g., Widmer 2002; Tobudic and Widmer 2003), in which the tempo of the performed pieces is not constant, and melody alterations are not permitted. (In classical music, melody alterations are often considered performance errors.) In those works, the focus is on global tempo and energy transformations, whereas we are interested in note-level timing and energy transformations as well as in melody ornamentations that are a very important expressive resource in jazz.

The induction of expressive-performance models using machine-learning techniques has also been applied to the identification of musicians from their playing styles. In this context, Saunders et al. (2004) applied string kernels to the problem of recognizing famous pianists from their playing styles. The characteristics of performers playing the same piece are obtained from changes in beat-level tempo and beat-level loudness. From such characteristics, general performance alphabets can be derived, and pianists' performances can then be represented as strings. They apply both kernel partial least squares and support vector machines to these data. Stamatatos and Widmer (2005) address the problem of identifying the most likely music performer, given a set of performances of the same piece by a number of skilled candidate pianists. They propose a set of simple features for representing stylistic characteristics of a music performer that relate to a kind of "average" performance. A database of piano performances of 22 pianists playing two pieces by Frédéric Chopin is used; they propose an ensemble of simple classifiers derived by both sub-sampling the training set and sub-sampling the input features. Experiments show that the proposed features are able to quantify the differences between music performers.

Ramirez et al. (2007) and Ramirez and Hazan (2007) investigate how jazz saxophone players express their view of the musical content of musical pieces and how to use this information to automatically identify performers. They study deviations of parameters such as pitch, timing, amplitude, and timbre both at an inter-note level and at an intra-note level. Their approach to performer identification consists of establishing a performer-dependent mapping of inter-note features (essentially a score, whether or not an actual score physically exists) to a repertoire of inflections characterized by intra-note features. They present a successful performer-identification case study.

Melodic Description

In this section, we describe how we extract a symbolic description from the monophonic recordings of performances of jazz standards and how we carry out a musical analysis based on the extracted symbolic description. 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 in this context.

Feature-Extraction Algorithms

First, we perform a spectral analysis of a portion of sound (the analysis frame), whose size is a parameter of the algorithm. This spectral analysis consists

of 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 msec, an overlap factor of 50%, and a Kaiser-Bessel 25-dB window.

Computation of Low-Level Descriptors

The main low-level descriptors used to characterize expressive performance are instantaneous energy and fundamental frequency. The energy descriptor is computed in the frequency domain, using the values of the amplitude spectrum at each analysis frame. In addition, energy is computed in different frequency bands as defined in Klapuri (1999), and these values are used by the algorithm for segmentation into notes.

For the estimation of the instantaneous fundamental frequency, we use a harmonic-matching model derived from the Two-Way Mismatch (TWM) procedure (Maher and Beauchamp 1994). 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 the absence of certain partials in the spectral data. The solution presented in Maher and Beauchamp (1994) 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 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 also has the benefit that the effect of any spurious components 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 earlier. Second, the prominent spectral peaks are detected. These spectral peaks are defined as the local maxima in the spectrum whose magnitudes are greater than a threshold. The spectral peaks are compared to a harmonic series, and a TWM error is computed for each fundamental-frequency candidate. 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 were added to deal with some of the algorithm's shortcomings. First, a peak-selection routine has been added 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. Second, we consider previous values of the fundamental frequency estimation and instrument dependencies to obtain a more adapted result. Finally, a noise gate based on some low-level signal descriptors is applied to detect silences, so that the estimation is only performed in non-silent segments of the sound.

Segmentation into Notes

Energy onsets are first detected following a band-wise algorithm that uses psychoacoustic knowledge (Klapuri 1999). In a second step, fundamental-frequency transitions are also detected. Finally, both results are merged to find the note boundaries (onset and offset information).

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 note's pitch and the fundamental

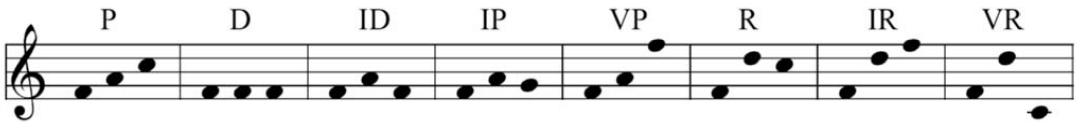


Figure 1. Prototypical Narmour structures. P = process; D = duplication; ID = intervallic duplication; IP = intervallic process; VP = registral process; R = reversal; IR = intervallic reversal; VR = registral reversal. For details, see Narmour (1990).

frequency that represents each note segment, as found in McNab, Smith, and Witten (1996). This is done to avoid taking into account mistaken frames in the fundamental-frequency mean computation. First, frequency values f are converted into cents c:

$$c = 1200 \frac{\log\left(\frac{f}{f_{ref}}\right)}{\log 2} \quad (1)$$

where $f_{ref} = 8.176$. Then, we define histograms with bins of 100 cents and a hop size of 5 cents, and we compute the maximum of the histogram to identify the note's pitch. Finally, we compute the mean frequency over all the points that belong to the histogram. The MIDI pitch is computed by quantization of this mean fundamental frequency over the frames within the note limits.

Musical Analysis

It is widely recognized that expressive performance is a multi-level phenomenon and that humans perform music considering a number of abstract musical structures. After computing the note descriptors as explained, and as a first step toward providing an abstract structure for the recordings under study, we decided to use Narmour's theory of perception and cognition of melodies (Narmour 1990) to analyze the structure of the music pieces performed.

The Implication/Realization model is a theory of melody perception and cognition. The theory states that a melodic musical line continuously causes listeners to generate expectations of how the melody should continue. An individual's expectations are motivated by two types of sources: innate and learned. According to Narmour, on one hand we are all born with innate information that suggests to us how a particular melody should continue. On the other hand, learned factors also influence our expectations and these factors are a result of exposure to music throughout our lives and our familiarity with musical styles and particular melodies.

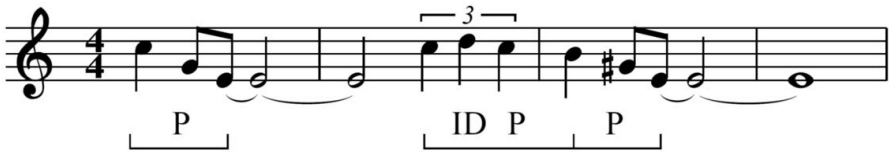
Any two consecutively perceived notes constitute a melodic interval, and if this interval is not conceived as complete, it is an *implicative interval,* that is, an interval that implies a subsequent interval with certain characteristics. That is to say, some notes are more likely than others to follow 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). The principle of intervallic difference states that a small interval (i.e., five semitones or less) implies a similarly sized interval (plus or minus two semitones), and a large interval (i.e., seven semitones or more) implies a smaller interval. 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. Thus, a description of a melody as a sequence of Narmour structures consists of a list of overlapping structures. Each melody in the training data is parsed to automatically generate an implication/realization analysis of the pieces. Figure 2 shows the analysis for a fragment of a melody.

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

Figure 2. Narmour analysis of All of Me.



aim is to obtain a model capable of automatically generating music performances with the expressiveness that characterizes human-generated music. In other words, 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).

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 eleven different tempi 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, the musician identified the fastest and slowest tempi at which each piece could be reasonably interpreted. Interpretations were recorded at regular intervals around the nominal tempo (five faster and five slower) within the fastest–slowest tempo limits. The data set is composed of 4,360 performed notes. Each note in the training data is annotated with its corresponding performed characteristics (i.e., performed duration, onset, and energy) and a number of score attributes representing both properties of the note itself and aspects of the context in which the note appears. Information about the note includes note duration and the note metrical position within a bar, and information about its melodic context includes performed tempo, information on neighboring notes, as well as the Narmour structure in which the note appears. (We focused on the Narmour group in which the note appears in third position, because this provides the best indicator of the degree to which the note is expected.)

Learning Task

In this article, we are concerned with note-level expressive transformations—in particular, transformations of note duration, onset, and energy. Initially, for each expressive transformation, we approach the problem as a classification problem: for note-duration transformations, for example, we classify each note as 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 to produce a numeric value representing the amount of transformation to be applied to a particular note. 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; `soft`, `loud`, and `same` for energy; and `ornamentation` and `none` for note alteration. A note is considered to belong to class `lengthen` if its performed duration is 20% longer (or more) than its nominal duration, that is, 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 as `lengthen`, for instance, was purposely lengthened by the performer and not the result of a performance inexactitude. A note is considered to belong to class `ornamentation` if a note or group of notes not specified in the score has been introduced in the performance to embellish the note in the melody, and to class `none` otherwise.

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 evolving a population of rules with the usual mutation and crossover operations. The algorithm constructs a hierarchical set of rules. Once constructed, the rules in the generated set are applied in the order they were generated. Thus, there is always a single rule that can be applied.

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 as negative examples the examples of the other two complementary classes (e.g., `shorten` and `same`).

In the case of note duration, onset, and energy, once we obtain the set of rules covering all the training examples, then for each rule, we apply linear regression to the examples covered by the rule to obtain a linear equation that predicts a numerical value. This leads to a set of rules producing a numeric prediction and not just a nominal class prediction. In the case of note alteration, we do not compute a numeric value; instead, we simply keep the set of examples covered by the rule. Later, for generation, we apply a standard k-nearest-neighbor algorithm to select one of the examples covered by the rule and adapt the selected example to the new melodic context (i.e., to transpose the ornamental note[s] to fit the melody key and ornamented note pitch). The algorithm is shown in Figure 3.

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. With 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. `NumericNewRule` is a rule in which the consequent `Regression(Rpos)` is a linear equation

$$X = w_0 + w_1 a_1 + w_2 a_2 + \ldots + w_k a_k \qquad (2)$$

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 alteration, namely, when dealing with ornamentations, `Regression(Rpos)` is simply the set examples covered by the rule.

Table 1. Parameter Values of the Genetic Algorithm

Parameter	*Identifier*	*Value*
Crossover rate	R	0.8
Mutation rate	m	0.05
Population size	p	200

The inner loop performs a finer-grained search to determine the exact form of each new rule. This is done by applying a genetic algorithm with the usual parameters r, m, and p, specifying the fraction of the parent population replaced by crossover, the mutation rate, and population size, respectively. The exact values for these parameters are presented in Table 1.

In the inner loop, a new generation is created as follows. First, probabilistically select $(1 - r)p$ members of P to add to the successor population Ps. The probability $Pr(h_i)$ of selecting hypothesis h_i from P is

$$\Pr(h_i) = \frac{Fitness(h_i)}{\sum h_j}, \ (1 \le j \le p) \qquad (3)$$

Next, probabilistically select $(r \times p)/2$ pairs of hypothesis from P (according to $Pr(h_i)$ above). For each pair, produce an offspring by applying the crossover operator (see subsequent description) and add it to the successor population Ps. Finally, choose the

Figure 3. Genetic sequential-covering algorithm used to train the expressive-performance model.

```
GeneticSeqCovAlg(Class,Fitness,Threshold,p,r,m,Examples)

        Pos = examples that belong to Class

        Neg = examples that do not belong to Class

        Learned_rules = {}

        While Pos do

                P = generate p hypotheses at random

                For each hypothesis h in P,

                        Compute fitness(h)

                While the highest fitness(h) in Pos less than Threshold do

                        Create a new generation Pnew

                        P = Pnew

                        For each h in P,

                                Compute fitness(h)

                NewRule = the hypothesis in P that has the highest fitness

                Rpos = members of Pos covered by NewRule

                Compute PredictedValue(Rpos)

                NumericNewRule = NewRule with Class replaced by Regression(Rpos)

                Learned_rules = Learned_rules + NumericNewRule

                Pos = Pos – Rpos

        Return Learned_rules
```

fraction *m* 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 the Narmour group by three bits. The last three

bits represent the predicted class (e.g., `shorten`, `same`, or `lengthen` for note duration).

Except for the Narmour group and the predicted class bits (i.e., the last six bits), two or more 1s in the bit string of a particular feature is interpreted as disjunction. For instance, the string `11010` for next-note duration means "the duration of the next note is either much shorter, shorter, or longer." (Note that in this context, `11111` is equivalent to `true`.) In the case of the Narmour-group bits, each string denotes a particular Narmour structure. Clearly, the predicted class string allows exactly one 1 representing the predicted class. (The genetic operators are designed in such a way that this is always the case.) 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 (referring to the score information) that 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. Both `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. Both `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 (an offbeat), or in none of these positions, respectively. Tempo is characterized as `slow`, `nominal`, or `fast` if the piece was performed at a speed slower than the nominal tempo (i.e., that identified as the most natural by the performer) by more than 15%, within 15% of the nominal tempo, or faster than the nominal tempo by more than 15%, respectively. In the case of the note's Narmour groups, we decided to code only one Narmour group for each note. That is, instead of specifying all the possible Narmour groups for a note, we select the one in which the note appears in third position.

Genetic Operators

We use the standard single-point crossover and mutation operators with two restrictions. To perform a crossover operation of two parents, the crossover points are chosen at random as long as they are on the attribute-substring boundaries. Similarly, the mutation points are chosen randomly as long as they do not generate inconsistent rule strings, for example, only one class can be predicted so exactly one 1 can appear in the last three-bit substring.

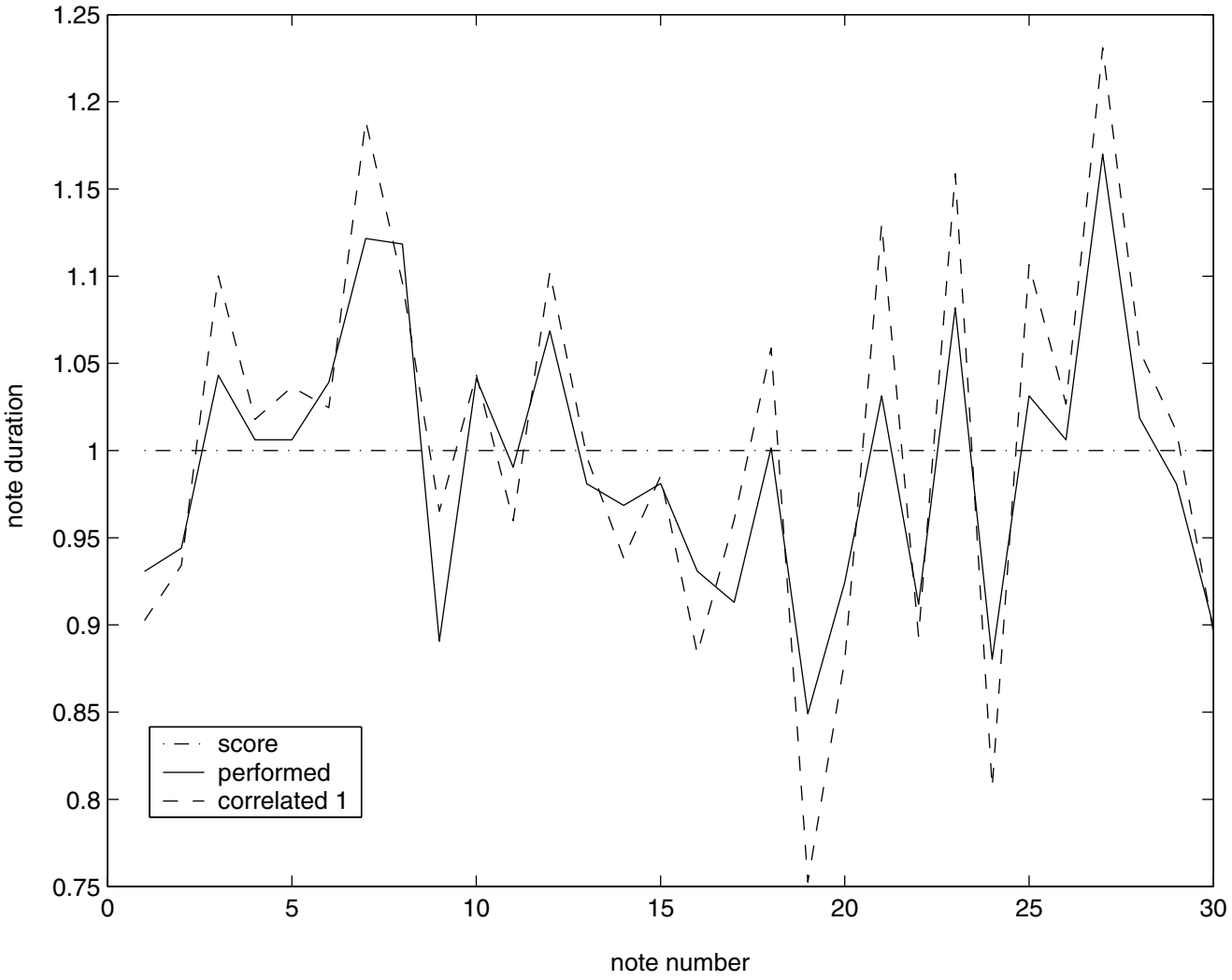
Fitness Function

The fitness of each hypothesized rule is based on its classification accuracy over the training data. In particular, fitness is defined as $t_p\alpha/(t_p + f_p)$, where t_p is the number of true positives, f_p is the number of false positives, and α is a constant that controls the true-positives to false-positives ratio. We set $\alpha = 1.15$, which, for our application, is a good compromise between coverage and accuracy.

Results

It is always difficult to formally evaluate a model that 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 can be evaluated by comparing the model's transformation predictions and the actual transformations performed by the musician. Figure 4 shows the note-by-note duration ratio predicted by a model induced by the algorithm and compares 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 4, the induced model seems to accurately capture the musician's expressive-performance

Figure 4. Comparison between model-predicted duration values and the actual performed values for the first 30 notes of Body and Soul *at a tempo of 65 beats per minute.*



transformations (despite the relatively small amount of training data).

The correlation coefficients for the onset, duration, and energy sub-models are 0.80, 0.84, and 0.86, respectively. These numbers were obtained by performing a ten-fold cross-validation on the data. At each fold, we removed the performances similar to the ones selected in the test set, that is, the performances of the same piece at tempi within 10% of performances in the test set.

We ran the sequential-covering genetic algorithm 20 times to observe the differences among the correlation coefficients from different runs. We observed no substantial differences. As a result of executing the genetic algorithm several times, we obtained different models. These models clearly share similar performance trends but at the same time generate slightly different expressive performances.

We allowed the user to influence the construction of the expressive model by imposing "readability constraints" on the shape of the rules. That is, the user was able to restrict the rule format (e.g., allow only some bit sequences) during the evolution to

enhance the interpretability of the induced rules. We examined some of the classification rules the algorithm induced (before replacing the class with the numerical predicted value), and we observed rules of different types. Some rules focus on features of the note itself and depend on the performance tempo, whereas others focus on the Narmour analysis and are independent of the performance tempo. Rules referring to the local context of a note (i.e., rules classifying a note solely in terms of the timing, pitch, and metrical strength of the note and its neighbors), as well as compound rules that refer to both the local context and the Narmour structure, were discovered.

To illustrate the types of rules found, we now present some examples of duration rules.

Rule 1

This rule, given by the sequence `11111 01110 11110 00110 00011 010 010 001`, states, "In 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."

Rule 2

This rule, given by the sequence `00111 00111 00011 01101 10101 111 111 100`, states "If the previous and next notes durations are longer (or equal) than the duration of the current note and the pitch of the previous note is higher, then shorten the current note."

Rule 3

This rule, given by the sequence `01000 11100 01111 01110 00111 111 111 010`, states, "If the previous note is slightly shorter and not much lower in pitch, and the next note is not longer and has a similar pitch (within a minor third), and the current note is not on a weak metrical position, then the duration of the current note remains the same (i.e., no lengthening or shortening)."

Analysis

These simple rules turn out to be very accurate: the first rule predicts 92%, the second rule predicts 100%, and the third rule predicts 90% of the relevant cases. Some of the rules turn out to be of musical interest; for instance, Rule 1 states that a note is to be lengthened if the two previous notes have the same pitch (i.e., it appears in a D Narmour group) and it has similar duration to the following note. This rule may represent the performer's intention to differentiate the last note of a sequence of notes with the same pitch.

Conclusion

This article 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 that predicts how a particular note in a particular context should be played (e.g., longer or shorter than its nominal duration). To induce the expressive-performance model, we extracted a set of acoustic features from the recordings resulting in a symbolic representation of the performed pieces, and we then applied a sequential-covering genetic algorithm to the symbolic data and information about the context in which the data appear. Despite the relatively small amount of training data, the induced model seems to accurately capture the musician's expressive-performance transformations. In addition, some of the classification rules induced by the algorithm proved to be of musical interest. Currently, we are in the process of increasing the amount of training data as well as experimenting with different information encoded in the data. Increasing the size of the training data set, extending the information in it, and combining it with background musical knowledge will certainly generate more models. We are also extending

our model to be able to predict intra-note expressive features such as vibrato and instantaneous energy. We characterize each performed note by its instantaneous pitch and energy, along with its timbre features, and we induce a model to predict these features according to the note's musical context.

Acknowledgments

This work is supported by the Spanish TIN Project ProSeMus (TIN2006-14932-C02-01). We would like to thank Emilia Gomez and Maarten Grachten for their invaluable help in processing the data, as well as the reviewers for their insightful comments and pointers to related work.

References

Biles, J. A. 1994. "GenJam: A Genetic Algorithm for Generating Jazz Solos." *Proceedings of the 1994 International Computer Music Conference.* San Francisco, California: International Computer Music Association, pp. 131–137.

Bilotta, E., P. Pantano, and V. Talarico. 2000. "Synthetic Harmonies: An Approach to Musical Semiosis by Means of Cellular Automata." In M. A. Bedau, et al., eds. *Proceedings of Artificial Life VII.* Cambridge, Massachusetts: MIT Press, pp. 537–546.

Dahlstedt, P., and M. G. Nordhal. 2001. "Living Melodies: Coevolution of Sonic Communication." *Leonardo* 34(3):243–248.

Degazio, B. 1999. "La Evolucion de los Organismos Musicales." In E. R. Miranda, ed. *Musica y Nuevas Tecnologias: Perspectivas para el Siglo XXI.* Barcelona: L'Angelot, pp. 137–148.

De Jong, K.A., et al. 1993. "Using Genetic Algorithms for Concept Learning." *Machine Learning* 13:161–188.

Friberg, A., et al. 1998. "Musical Punctuation on the Microlevel: Automatic Identification and Performance of Small Melodic Units." *Journal of New Music Research* 27(3):217–292.

Grachten, M., J. Luis Arcos, and R. Lopez de Mantaras. 2004. "Evolutionary Optimization of Music Performance Annotation." *Proceedings of the 2004 Conference on Computer Music Modeling and Retrieval.* Berlin: Springer, pp. 347–358.

Hazan, A., et al. 2006. "Modeling Expressive Performance: A Regression Tree Approach Based on Strongly Typed Genetic Programming." *Proceedings of the European Workshop on Evolutionary Music and Art.* Berlin: Springer, pp. 676–687.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems.* Ann Arbor, Michigan: University of Michigan Press.

Horner, A., and D. E. Goldberg. 1991. "Genetic Algorithms and Computer-Assisted Music Composition." *Proceedings of the 1991 International Computer Music Conference.* San Francisco, California: International Computer Music Association, pp. 479–482.

Hunt, A., R. Kirk, and R. Orton. 1991. "Musical Applications of a Cellular Automata Workstation." *Proceedings of the 1991 International Computer Music Conference.* San Francisco, California: International Computer Music Association, pp. 165–166.

Klapuri, A. 1999. "Sound Onset Detection by Applying Psychoacoustic Knowledge." *Proceedings of the 1999 IEEE International Conference on Acoustics, Speech and Signal Processing.* Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, pp. 3089–3092.

Lopez de Mantaras, R., and J. L. Arcos. 2002. "AI and Music: From Composition to Expressive Performance." *AI Magazine* 23(3):32–57.

Madsen, S. T., and G. Widmer. 2005. "Exploring Similarities in Music Performances with an Evolutionary Algorithm." *Proceedings of the International FLAIRS Conference.* Menlo Park, California: AAAI Press, pp. 80–85.

Maher, R. C., and J. W. Beauchamp. 1994. "Fundamental Frequency Estimation of Musical Signals Using a Two-Way Mismatch Procedure." *Journal of the Acoustical Society of America* 95(4):2254–2263.

Mandelis, J. 2001. "Genophone: An Evolutionary Approach to Sound Synthesis and Performance." In E. Bilotta, et al., eds. *Proceedings of ALMMA 2002 Workshop on Artificial Models for Musical Applications.* Castrolibero, Italy: Editoriale Bios, pp. 108–119.

Manzolli, J., et al. 1999. "An Evolutionary Approach Applied to Algorithmic Composition." In E. R. Miranda and G. L. Ramalho, eds. *Proceedings of the VI Brazilian Symposium on Computer Music.* Rio de Janeiro: SBC/Entre Lugar, pp. 201–210.

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

McNab, R. J., L. A. Smith, and I. H. Witten. 1996. "Signal Processing for Melody Transcription." *Proceedings of the 19th Australasian Computer Science Conference.* Melbourne, Australia: University of Melbourne and RMIT, pp. 301–307.
Michalski, R. S. 1969. "On the Quasi-Minimal Solution of the General Covering Problem." *Proceedings of the First International Symposium on Information Processing,* Bled, Yugoslavia: N.P., pp. 125–128.
Millen, D. 1990. "Cellular Automata Music." *Proceedings of the 1990 International Computer Music Conference.* San Francisco, California: International Computer Music Association, pp. 314–316.
Miranda, E. R. 1993. "Cellular Automata Music: An Interdisciplinary Music Project." *Interface* 22(1):3–21.
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.
Narmour, E. 1990. *The Analysis and Cognition of Basic Melodic Structures: The Implication Realization Model.* Chicago: University of Chicago Press.
Ramirez, R., et al. 2005. "Understanding Expressive Transformations in Saxophone Jazz Performances." *Journal of New Music Research* 34(4):319–330.
Ramirez, R., et al. 2006. "A Data Mining Approach to Expressive Music Performance Modeling." In Valery Petrushin, ed. *Multimedia Data Mining and Knowledge Discovery.* Berlin: Springer, pp. 362–380.
Ramirez, R., and A. Hazan. 2005. "Understanding Expressive Music Performance Using Genetic Algorithms." *Proceedings of the European Workshop on Evolutionary Music and Art.* Berlin: Springer, pp. 508–516.
Ramirez, R., et al. 2007. "Performance-Based Interpreter Identification in Saxophone Audio Recordings." *IEEE Transactions on Circuits and Systems for Video Technology* 17(3):356–364.
Ramirez, R., and A. Hazan. 2007. "A Rule-Based Expressive Performance Model for Jazz Saxophone." *Proceedings of the International Workshop on Artificial Intelligence and Music.* Hyderabad, India: IAAA, pp. 37–42.
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* 92(5):2546–2568.
Saunders, C., et al. 2004. "Using String Kernels to Identify Famous Performers from Their Playing Style." *Proceedings of the 15th European Conference on Machine Learning.* Berlin: Springer, pp. 2546–2568.
Stamatatos, E., and G. Widmer. 2005. "Automatic Identification of Music Performers with Learning Ensembles." *Artificial Intelligence* 165(1):37–56.
Tobudic, A., and G. Widmer. 2003. "Relational IBL in Music with a New Structural Similarity Measure." *Proceedings of the International Conference on Inductive Logic Programming.* Berlin: Springer, pp. 37–56.
Todd, N. 1992. "The Dynamics of Dynamics: A Model of Musical Expression." *Journal of the Acoustical Society of America* 91:3540–3550.
Waschka II, R. 1999. "Avoiding the Fitness Bottleneck: Using Genetic Algorithms to Compose Orchestral Music." *Proceedings of the 1999 International Computer Music Conference.* San Francisco, California: International Computer Music Association, pp. 201–203.
Widmer, G. 2002. "Machine Discoveries: A Few Simple, Robust Local Expression Principles." *Journal of New Music Research* 31(1):37–50.