Evolving Musical Performance Profiles Using Genetic Algorithms with Structural Fitness

Qijun Zhang Eduardo Reck Miranda
Computer Music Research
School of Computing, communications and Electronics
University of Plymouth, UK
Drakes Circus, PL4 8AA
(+44) 1752 232579
{qijun.zhang, eduardo.miranda}@plymouth.ac.uk

ABSTRACT
This paper presents a system that uses Genetic Algorithm (GA) to evolve hierarchical pulse sets (i.e., hierarchical duration vs. amplitude matrices) for expressive music performance by machines. The performance profile for a piece of music is represented using pulse sets and the fitness (for the GA) is derived from the structure of the piece to be performed; hence the term "structural fitness". Randomly initiated pulse sets are selected and evolved using GA. The fitness value is calculated by measuring the pulse set's ability of highlighting musical structures. This measurement is based upon generative rules for expressive music performance. This is the first stage of a project, which is aimed at the design of a dynamic model for the evolution of expressive performance profiles by interacting agents in an artificial society of musicians and listeners.

Categories and Subject Descriptors
I.2 [**Artificial Intelligence**]: General – *Cognitive simulation*. J.5 [**Arts and Humanities**]: Performing arts.

General Terms
Algorithms, Design, Experimentation, Human Factors.

Keywords
Application, art and music, entertainment and media.

1. INTRODUCTION
Music performances with proper expressions are defined as expressive music performances. "Proper expressions" is what makes music interesting and sound alive. In the context of Western tonal music, there is a commonly agreed notion that expression is conveyed in a music performance by delicate deviations of the notated musical score during the performance. Thus, expressive music performance research is aimed at establishing why, where and how these deviations take place in a piece of music. Interestingly, even though there are many commonalities in musical performance practices, these deviations can vary substantially from performance to performance, even when the same performer plays the same piece of music more than once. This is one of the main reasons for employing Genetic Algorithms (GA) to evolve performance profiles. This rationale will become clearer as this paper develops.

One of the objectives of designing computational models of expressive performance is to connect the properties of a musical score and performance context with the physical parameters of a performance, such as timing, loudness, tempo, articulation and so on. These models help us to gain a better understanding of expressive music performance and provide technology to implement systems to perform music. Different strategies have been employed in expressive performance research (e.g., analysis-by-measurement, analysis-by-synthesis, machine learning and so on) in order to capture common performance principles. Comprehensive reviews about these works can be found in [7, 10].

As a matter of fact, social factors, including the influence of historical practices and the interactions between performers and audience, play an important role in music performance [8]. However, the frequently used strategies can help little to investigate this aspect. Therefore, the aim of our research is to build an evolutionary simulation model that takes into account these social factors by simulating the interactions among performers and listeners, through which expressive music performance profiles emerge as a result of musical constraints and social pressure.

This paper presents the first stage of our project, which is a GA-based system that evolves performance profiles with fitness rules derived from musical constraints; in this case, from the structure of the pieces to be performed. The system evolves suitable performance profiles from randomly initiated ones using GA combined with generative rules of expressive music performance [2]. We apply hierarchical pulse set to represent performance profiles, which define deviations for note duration and amplitude values when playing a piece of music (in MIDI format). The fitness value of a pulse set is calculated according to several rules derived from the research into perception of musical structure. Rather than directly constructing performance profile with these rules, GA is used to search for suitable pulse sets to perform a piece.

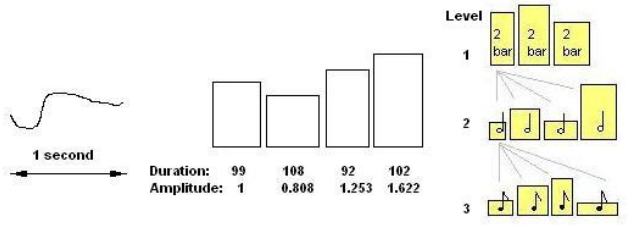
Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
GECCO'06, July 8–12, 2004, Seattle, Washington, USA.
Copyright 2006 ACM 1-59593-186-4/06/0007…$5.00.

2. MUSICAL PERFORMANCE WITH HIERARCHICAL PULSE SET

In this section we introduce the notion of pulse sets, and how we use them as performance profiles to perform musical pieces.

2.1 Notion of pulse set

Figure 1.a shows a pulse represented as a curve of measurements of finger pressure on a pressure sensor pad. The information in a pulse is a wrap of specific temporal patterns with amplitude patterns, and can be quantified as real numbers (width and height correspond to duration and amplitude, respectively), as depicted in Figure 1.b. A pulse can operate at different levels of temporal organization and can be grouped into a hierarchical structure [3]. Manfred Clynes proposed to represent a hierarchical pulse set as a matrix of duration and amplitude values (shown in figure 1.c), which defines the deviations of the physical attributes of musical notes. This makes it possible to generate computer performance, by modulating the physical attributes of musical notes according to these deviations.



a. **b.** **c.**

Figure 1. Illustration of a pulse and the notion of hierarchical pulse sets. (a) A pulse represented as finger pressure measurements in time. (b) A representation of pulse as a wrap of real numbers (duration vs. amplitude). (c) A hierarchical pulse set derived from grouping pulses.

2.2 Pulse sets as performance profiles

We adopted the notion of hierarchical pulse sets to represent performance profiles in our work for three reasons. Firstly, the choices of notes' duration and amplitude significantly influence the expressive quality of a music performance, although not fully. Secondly, the hierarchical nature of pulse sets matches important features of most music genres; e.g., the notions of grouping and hierarchical structures. Finally, we regard hierarchical pulse sets as rather compact and informative forms for generative music interpretation.

Table 1 shows an example of a hierarchical pulse set and its components' meanings. This example is the quantification of the pulse set drawn in Figure 1.c. We briefly explain this example below and also introduce the values' ranges.

2.2.1 Representation of a pulse set

In the first line, 8 defines that the smallest unit of the music to be operated on is the eighth note. The other possible values here could be 32, 16 or 4. That is, in the present version of the system, the shortest note can be a thirty-second note, a sixteenth note, an eighth note or a quarter note.

In this example, there are 4, 4 and 3 elements in each level, respectively, from Level 3 (lowest) to Level 1. All the elements in a lower level are part of one element in its upper level. Therefore, as depicted in Figure 1.c, an element in Level 2 equals to the total length of Level 3, that is 4×eighth notes=a half note. Similarly, one unit in Level 1 has the entire length of all elements in Level 2. Assuming that there are four beats in each bar, then this pulse set defines three 2-bar groups. In our system, the number of elements in one level is valid if it is an integer higher than 2 and lower than 9 (2 and 9 are inclusive).

Since a hierarchical pulse set informs the deviation of durations and amplitudes of notes, this information is given from the third to the last line in the representation of a pulse set. In our system, the duration value can be any integer between 75 and 125, and the amplitude value varies from 0 to 1.5.

Table 1. Representation of pulse set and explanation.

Pulse set example	Meaning
8	The length of note at the lowest level
4 4 3	Number of elements in three levels (From the lowest level to the highest)
0.339 0.762 0.953 0.319	Level 3 Amplitude (lowest level)
73 93 66 124	Level 3 Duration
0.453 0.798 0.498 1.333	Level 2 Amplitude
62 103 114 118	Level 2 Duration
1.398 1.476 1.864	Level 1 Amplitude
73 121 120	Level 1 Duration

2.2.2 Calculating a deviation pattern from a pulse set

As we explained earlier, the pulse set example in Table 1 defines a performance profile for a 6-bar segment. Defined by its hierarchical feature, there are 48 (4×4×3) pulse elements that together compose the segment. The deviation pattern is refined for each one of them. The duration and amplitude for each element are calculated in a top down manner, by multiplying the parameters of corresponding elements in different hierarchical levels. For instance, the 1st and the 40th pulse element (represented as e1, e40) in this list are defined by the following elements of the pulse set, respectively:

$\mathbf{e_1}$: the 1st in Level 1, the 1st in Level 2, the 1st in Level 3
$\mathbf{e_{40}}$: the 3rd in Level 1, the 2nd in Level 2, the 4th in Level 3

According to the parameters of the pulse set, the algorithm then calculates the duration and amplitude values for these two pulse elements, as shown in Table 2.

Table 2. Calculation for a pulse element in a pulse set.

Note	Duration	Amplitude
$\mathbf{e_1}$	$73\times62\times73 / 100^3$	$1.398\times0.453\times0.339$
$\mathbf{e_{40}}$	$120\times103\times124 / 100^3$	$1.864\times0.798\times0.319$

With this method, we can draw deviation patterns, or envelopes, for both duration and amplitude values. Once started, these patterns repeat until the piece finishes. For the sake of clarity, Figure 2 shows only the first half of a deviation pattern based on

the example pulse set. The index of the beat in the piece is given by the x axis, while y corresponds to the calculated percentage deviation of duration or amplitude.

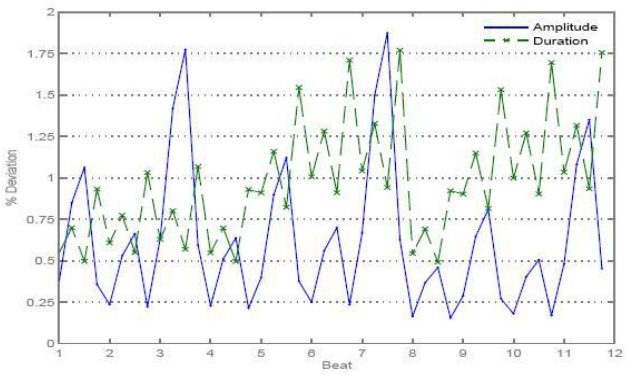


Figure 2. Deviation pattern for the pulse set in Table 1.

2.3 Implementation issues

The musical pieces that were used to test our system were stored as flat MIDI files, which don't have any timing deviation (i.e., the rhythm is exactly as written on the score) and with even (i.e., equal) loudness for all notes.

The performance of a piece proceeds as follows: firstly, we look up its start time in the aforementioned deviation list to infer its position along with its detailed duration and amplitude. Inspired by a method proposed by Manfred Clynes [4], a note's playing time is given by summing all the durations of the pulse components (i.e., if this note is longer than the smallest unit), while the amplitude is defined by the amplitude information of its first pulse component. Precisely speaking, when we modify the duration of a note in a MIDI file, we change the file's play back tempo at the required positions. We reassign a note's "note-on velocity" MIDI code value to modify its amplitude.

Through the above modification, the system produces a new MIDI file with added expressions. Then, this is evaluated according to the performance principles introduced below.

3. FITNESS FUCTION BASED ON MUSICAL STRUCTURE

It is commonly agreed that there is a strong relation between expression in music performance and music structure [9]. This is an important reason for the existence of commonalities in different performers' performances of the same piece, and thus a necessary hypothesis for modeling expressive music performance. Those using the analysis-by-synthesis approach have built models loaded with comprehensive rules. However, a critical problem of these rule-based systems is the way in which they combine these rules [5]. They can be combined in many different ways and most combinations can generate conflicting situations with no objective solution. The reality is that there are different ways to perform a piece of music, which make it very difficult to fully formalize musical performance with rules.

We have no desire to compile a comprehensive collection of fixed performance rules manually, not only because of the above problem, but also because this is not the main point of our research. Rather, we are interested in a system that can evolve these rules dynamically. Nevertheless, we undoubtedly need guidance from musically meaningful rules (e.g., musical structure, etc.) to evaluate whether a pulse set works well for a certain performance profile.

For this purpose, our approach is to design descriptive performance principles without quantified regulations. And then we employ GA, whose fitness function is informed by these principles, to select and evolve suitable pulse sets, starting from randomly generated sets. In this sense, the usage of GA is ideal here because otherwise it would be hard to design manually a decent performance profile based on such descriptive principles. Furthermore, GA can evolve different and suitable pulse sets for the same piece. This diversity is a noticeable phenomenon in real performances, and also a prerequisite for the next stage of our research.

3.1 Structure Analysis

In order to use any structural principles for calculating fitness values, firstly we need to analyze the structure of piece in question. In the present version of our system, we use David Temperley's software Melism, which performs several structural analysis such as metrical analysis, group analysis, harmony analysis and key analysis [9].

3.2 Selected performance principles

Our descriptive performance principles are very much inspired by Eric Clarke's generative rules for expressive performance [2]. We associate expressions in performance with the piece's structure features of grouping, accentuation and cadence. Thus, the fitness value of a pulse set consists of three parts, FitGrouper, FitAccent and FitCadence.

3.2.1 FitGrouper

FitGrouper is obtained by a pulse set's fitness in relation to two rules, mainly concerning the notes' duration at group boundaries. These two rules are:

Rule 1: The time deviation of the last note of a group has either larger or smaller timing deviation than both the notes before and after it.
Rule 2: The last note of a group is always lengthened in order to delay the following note and signify the starting of a new group.

The value of *FitGrouper* is dependent on a pulse set's violation of above two rules. A parameter *numVio* (initialised equal to 0) increases whenever the pulse set breaks either Rule 1 or Rule 2. If the number of groups in the piece is N_{group}, we define $FitGrouper = 1 - \frac{numVio}{N_{group}}$. The maximum value of *FitGrouper* is equal to 1.

3.2.2 FitAccent

FitAccent is an evaluation of how well the notes' loudness contour in an "interpreted" piece (i.e., after the flat MIDI file is modulated by a given pulse set) fits the metrical analysis. The rule used here is as follows:

Rule 3: Preference should be given to the contour of notes' loudness that has the most similar shape to the music's accentuation analysis.

Given two successive notes N0 and N1, *FitAccent* is produced by calculating:

(i) The accentuation information (b0, b1) from the structure analysis, and
(ii) The Velocity information (v0, v1) from the interpreted MIDI file.

Because the accent value b_i varies from 0 to 4, firstly, we normalize the velocity difference (v1-v0) to integers in the range of [-4, 4]. Then we assign a reward value between 0 and 1 to parameter x based on the difference between (v1-v0) and (b1-b0). The closer they are to each other, the larger the value assigned to x. If the number of notes in the piece is N_{note}, we define $FitAccent == \frac{\sum_{1}^{N_{note}-1} x_i}{N_{note}-1} (0 \le x_i \le 1)$. As with *Fit Grouper,* the maximum value of *FitAccent* is equal to 1.

3.2.3 FitCadence

FitCadence takes into account the strong chord progressions in a piece, which can also indicate group boundaries. While both *FitGrouper* and *FitAccent* work at the note level, *FitCadence* judges the performance features of a higher group level. The rule for calculating *FitCadence* is as follows:

Rule 4: Both segments corresponding to two chords in a cadence (e.g., V→I, IV→ I, or Dominant→Tonic, Subdominant→Tonic, respective) should be lengthened. Different weights are set for different categories of cadences because they have varying importance on a piece's structure.

As with the *FitGrouper*, the value of *FitCadence* is also decided by a pulse set's violation of Rule 4. And the pulse set will receive more penalties when it breaks the rule with stronger cadences. If the number of cadences in a piece is $N_{cadence}$, and we assign weight w_i to the i^{th} cadence, then

$FitCadence = 1 - \frac{\sum_{1}^{Ncadence} w_i}{N_{cadence}}$. Wit with the previous two fitness measures, the maximum value of *FitCadence* is equal to 1.

In the present version of our system, we define the total fitness of a pulse set to be the sum of *FitGrouper*, *FitAccent* and *FitCadence*. That is, Fitness = FitGrouper + FitAccent + FitCadence, with maximum value equal to 3.

4. EVOLUTION PROCEDURE

It this section we introduce the procedure to evolve suitable pulse sets from scratch.

4.1 Genome representation of a pulse set

A pulse set is represented by a long string of real numbers in the same order as shown in Table 1. In this string, we separate lines with ";" and insert "," between elements in the same line. This makes it convenient to access and operate on parameters of different hierarchical levels. An additional number, either 0 or 1, is added at the end of an individual pulse set. This is used to indicate one of the possible two ways of applying a crossover operation, which will be clarified later.

As an example, the pulse set in Table 1 is represented as follows (for the sake of clarity, we omitted Level 2 and Level 1); the additional number at the end of the string is equal to 0:

8;4,4,3;0.339,0.762,0.953,0.319;73,93,66,124; … ;0

4.2 Initialization of the first generation

The individual pulse sets of the first generation are randomly generated. For the moment, we have established that all pulse sets have 3 levels. All other pulse set values are randomly generated, including:

(1) The length of quickest note

(2) The number of elements in each hierarchical level

(3) Amplitude and duration values for each element in every level

(4) The additional number at the end of the string (for selecting the crossover operation)

4.3 Evolution algorithm

For every generation, each pulse set is used to modulate the flat MIDI file, as described in section 2, and a fitness value is calculated according to the definition of fitness functions introduced in section 3. Thus, we obtain an array of values $Fit0=f_1,f_2,\ldots,f_n$, where f_i is the fitness value of the i^{th} individual pulse set. The offspring pulse sets for the next generation are created on the basis of this fitness array. The procedure is as follows:

(1) Calculate the fitness values of the current generation P0

(2) Select parent candidates to compose of population $P0_1$

(3) Operate mutation on $P0_1$ and get population $P0_2$

(4) Operate crossover on pairs of pulse sets in $P0_2$ to get population $P0_3$

(5) Rank the fitness values of Generation P0 and $P0_3$ and the best half become generation P1

(6) Repeat the steps from (1) to (5) until completing a preset number of generations.

4.4 Genetic operations

In this section we explain the three genetic operations used in the evolution procedure: selection, mutation and crossover, respectively.

4.4.1 Selection

Based on Tobias Blickle's [1] comparative study of various widely used selection operators in GA (such as, tournament, linear and exponential rankings, and proportional), we opted for using exponential ranking. This is because we wish to keep a certain degree of diversity in the evolutionary process and exponential ranking has proved to work well for this purpose. The pseudo-code of our exponential ranking selection is as follows:

Exponential-ranking(c, J_1, … , J_n)

J←sorted population J according to fitness (first is the worst)
S0←0
For i←1 to N do
s_i←S_{i-1} + p_i
Do

For i←1 to N do
r←random[0,s_N]
J_i←J_k such that $s_{i-1} \le r < s_k$
Do
Return

Here, the value of c is randomly generated every generation from 0.75 to 1.

4.4.2 Mutation

Considering the hierarchical property of a pulse set, we have employed four different ways for operating mutation on a single pulse set. Given a pulse set, Figure 3 shows examples of how each of the following mutation schemes work: Ma, Mb, Mc and Md.

Ma: Randomly modify every duration or amplitude values in the pulse set. The range of changes for amplitude is [-0.1, 0.1], and for duration is [-5, 5].

Mb: Append new duration and amplitude wraps or delete existing wraps from the end of the string. The number of added or removed elements is defined randomly, with the condition that the resulting pulse set is a valid pulse set. Also, if new elements are added, they are generated randomly. The length of the quickest note in the pulse set may be changed in this mutation, as we generate a new value for it randomly.

Mc: Swap the order of elements in the same level of the pulse set randomly, but keep the duration and amplitude wraps.

Md: Swap the order of hierarchical levels in the pulse set randomly.

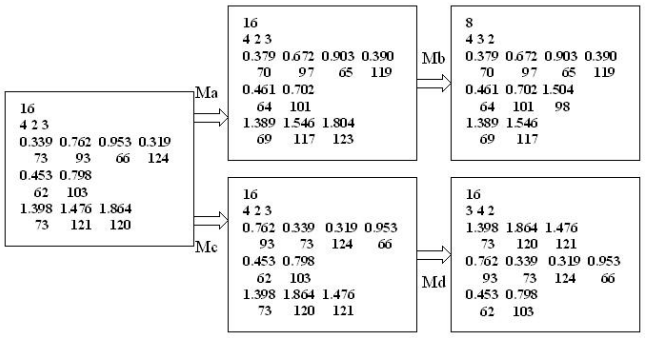


Figure 3. Examples of mutation schemes.

An integer between 1 and 4 is generated randomly in each generation. This number defines which mutation methods can be used. For example, if we obtain 2, then only the first two mutation methods (Ma and Mb) are used in the respective generation. Then, whether to perform Ma or Mb to an individual pulse set is also decided randomly.

4.4.3 Crossover

To maintain the evolved parameters in hierarchical levels, we only allow for segmentation and crossover at the positions of complete hierarchical levels. Therefore, crossover enables two parents pulse sets to exchange some of their component levels. For example, let us consider the crossing over of two pulse sets: P_x and P_y. They can be respectively represented as $X_1X_2X_3x$ and $Y_1Y_2Y_3y$, where X_n or Y_n refers to the nth level of P_x or P_y, including duration and amplitude parameters. The variable x or y represents the number at the end of P_x or P_y, which can value either 0 or 1.

Here we use the value of x-y (which can be 0, 1, or -1) to decide the way in which to perform crossover with P_x and P_y. This includes the choice of one-point crossover or two-point crossover, as well as which levels of the parent pulse sets the crossover will operate on. Possible crossovers are shown in Table 2. If x equals to y through crossover, then P_x and P_y exchanges their middle level, keeping all other information unchanged. If x=0 and y=1, then P_x and P_y exchange their lowest level along with the last number. Otherwise, if x=1 and y=0, then the highest level of P_x and P_y are crossed over.

Table 3. Crossover scheme.

x \ y	0	1
0	$X_1Y_2X_3x$ $Y_1X_2Y_3y$	$X_1X_2Y_3y$ $Y_1Y_2X_3x$
1	$X_1Y_2Y_3y$ $Y_1X_2X_3x$	$X_1Y_2X_3x$ $Y_1X_2Y_3y$

5. DEMONSTRATION

As a demonstration, we present an example with the melody of Robert Schumann's Träumerei. Figure 4 shows the structural analysis used for calculating the fitness value, including grouping structure, metrical analysis and harmonic progression. Group boundaries are noted by "xx", vertically positioned under the staves at segmenting positions. The numbers at the bottom of the notes correspond to accent information (from metrical analysis). The chord names above the staves indicate chord progressions.

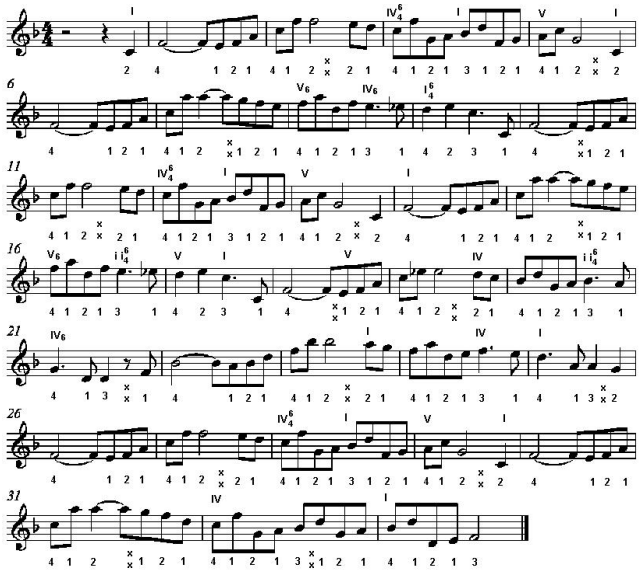


Figure 4. The soprano part of Robert Schumann's Träumerei and its structural analysis.

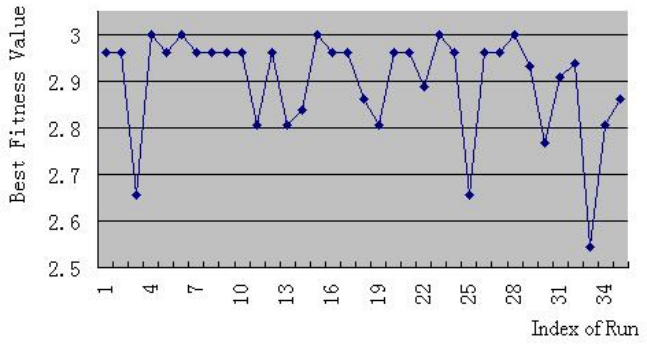


Figure 5. The best fitness value of 35 runs.

The results discussed below were generated by running the evolution procedure 35 times. In each run, 100 individual pulse sets are randomly generated for the first generation, and then we let them evolve for 100 generations.

For each run, we have recorded for every each generation: a) every pulse set's fitness values including FitGrouper, FitAccent, FitCadence and the total fitness (i.e., the sum of the three) and b) the best pulse set. Figure 5 depicts the final best fitness values that we recorded from all runs; that is, these are the best fitness values of generation number 100 for each of the 35 runs. As shown in Figure 5, "excellent" pulse sets whose fitness values reached 3.0 have evolved in the 4th, 6th, 15th, 23rd and 28th runs. This does not necessarily mean that each of these runs produced only one "excellent" pulse set each. In fact, each run produced more than one. Although most of these "excellent" pulse sets may share identical configurations (which is a pulse set's basic structure decided by the first two lines of its representation), they always have different duration and amplitude parameters, resulting in different performance profiles. For example, the system evolved two "excellent" pulse sets with two different configurations in the 15th run. Those "excellent" pulse sets from different runs are obviously different from each other. We list some examples in Table 4: two pulse sets evolved in the 4th run and two evolved in the 15th run.

Table 4. Example of "excellent" pulse set.

Run4_1	**Run15_1**
8 8 2 2 1.464 0.767 0.925 0.15 1.262 0.622 1.025 0.388 97 86 116 107 123 106 60 113 1.046 1.004 95 115 1.282 1.331 118 121	16 4 4 2 1.382 0.573 0.676 0.109 125 116 100 80 1.369 0.875 1.107 0.883 116 115 109 80 1.133 1.116 115 122
Run4_2	**Run15_2**
8 8 2 2 1.478 0.582 1.036 0.344 1.301 0.305 1.022 0.606 101 84 121 101 123 113 54 101 1.057 1.05 100 115 1.393 1.348 124 121	16 4 4 3 1.418 0.598 0.525 0.232 122 120 102 90 1.336 0.853 1.087 0.87 116 123 111 88 1.154 1.113 1.096 118 122 110

From Table 4 we can infer that both pulse sets evolved in Run 4 have a repeated deviation pattern that consists of eighth_note×8×2×2=16 beats, which corresponds to the length of 4 bars. The other two pulse sets evolved in Run 15 have different configurations: Run 15_1 is for a 2-bar period (sixteenth_note×4×4×2=8 beats) and Run 15_2 is for a 3-bar period (sixteenth_note×4×4×3=12 beats). Figure 6 depicts their deviation pattern in 8 beats (2 bars).

At present, we judge if a pulse set is suitable for the piece mainly depending on how well they fulfill the devised rules for the fitness function. This can be done by checking whether the important notes' deviations correspond to those described in the rules. Here we give a brief analysis taking the examples in Figure 6. Firstly, we can list the group boundaries in the piece based on the score in Figure 4. They are the notes at the 10th, 18th, 26th, 37th beat and so on, always taking more than one beat. It is not hard to find out from Figure 6.a that each of these notes' duration (adding all the beats occupied by each note) deviates mostly compared with its two neighbor notes. In this way, it is fair to say that both Rule 1 and Rule 2 have been satisfied because all notes were lengthened. Secondly, for Rule 3, as it is the only rule affecting notes' amplitude in our present system, we can see in Figure 6.b that all those "excellent" pulse sets follow identical amplitude deviation patterns, which match the accentuation information shown in the score. Finally in terms of Rule 4, there are several cadences such as V→I, IV→ I in this piece. We have checked that in most of the cases, both groups of notes composing the two chords of a cadence were lengthened. Other violations might have occurred because the concurrent effect of Rule 1 and Rule 2.

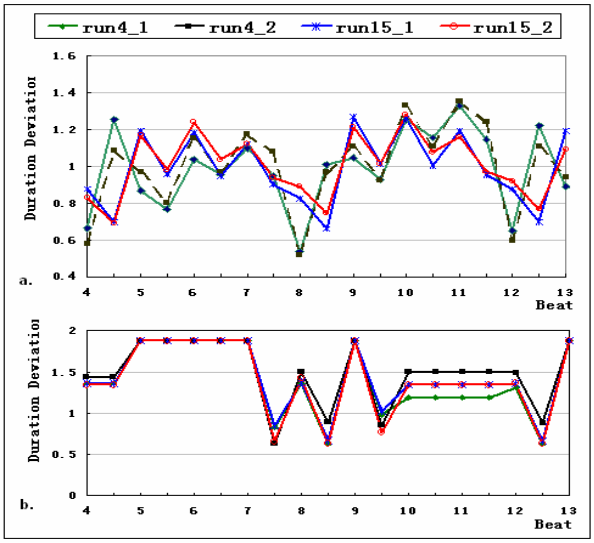


Figure 6. Examples of deviation pattern by evolved pulse sets. It starts from the 4th beat because the piece actually begins with an upbeat at the 4th beat.

In another analysis, we have drawn four curves for each run, for the values of *FitGoruper*, *FitAccent*, *FitCadence* and the total fitness, respectively, in order to follow the development of fitness values through 100 generations. This was done according to the corresponding data of the best pulse set in each generation. Figure 7 shows the geometric means of these 35 groups of curves accordingly. These fitness curves show how the best pulse sets changed through the generations. We can observe that FitGrouper and FitCadence, which are fitness components defined as penalties for breaking the rules, have always played a dominant role in the beginning of the evolution. It is also possible to observe that after both of them have reached the maximum value 1, the configuration of the best pulse set in the following

generations had hardly changed. During this sustain period, the best pulse set gradually spreads over the population, which indicates convergence, even though modifications on the duration or amplitude parameters kept taking place. Although this is a dominant development, it is not absolute because there still is the possibility that some exceptionally good configuration had emerged; a good example of this are the pulse sets in the 15th run.

We also have done some other experiments to observe the effects of mutation on the best fitness value that pulse sets can have. For example, by adding the step to randomly generate a new value for the quickest note of every mutation scheme, we found that it is hard to evolve pulse sets with fitness value as high as 3.

The mutation scheme Ma is always performed in the present version of the system. Although this has been decided on purpose, it would be interesting to observe what happens if we change the order of the mutation schemes.

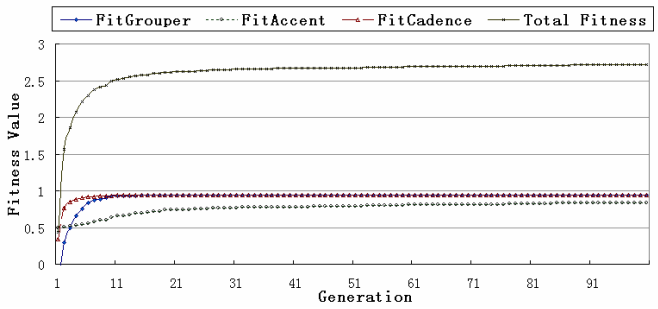


Figure 7. Fitness trace. This is achieve by averaging the fitness value of the evolved best pulse set in the same generation across 35 runs

6. CONCLUDING REMARKS

In this paper we introduced a novel application of GA: to evolve music performance. GA evolves suitable pulse sets for musical performance using fitness rules derived from the structure of the piece to be performed. Furthermore, the "excellent" pulse sets evolved by the GA, no matter whether they were from the same run or not, have shown diversity and also commonality. This could be observed both objectively (by comparing the figures of deviation patterns by different pulse sets) and subjectively (by listening to the "interpreted" MIDI files).

When listening to pieces performed with the evolved pulse sets, we can perceive the expressive dynamics of the piece, mainly due to lengthening or shortening of related notes. However, we acknowledge that such subjective assessment of the results does not hold much scientific value. We are currently developing methodology to validate the evolved pulse sets in comparison with human performanced. The study of Bruno Repp [5] is a very helpful resource for this purpose.

We are currently testing the systems with different settings and variations with the objectives of gaining a better understanding of its behavior and fine-tuning it for the next stage of our research with multiple interactive agents. Other ongoing work includes:

(1) We are taking into account performance principles associated to or determined by melody. Melodic information will improve the grouping and accentuation analysis.

(2) We are implementing a mechanism to vary the number of hierarchical levels in order to render the model more robust when it encounters more complex music structures. We feel that sometimes the model would benefit from being able to cope with more hierarchical levels when evolving pulse sets for pieces of higher complexity than the pieces we have tested so far.

(3) We are devising a new way to compute the fitness function, as a weighted sum of the fitness values for different performance principles. We are interested in letting these weights to evolve with the pulse sets.

7. REFERENCES

[1] Blickle, T. "A comparison of selection schemes used in genetic algorithms". Technical report, Computer Engineering and Communication Networks Lab (TIK), Swiss Federal Institute of Technology (ETH) Zurich (1995).

[2] Clarke, E. F. "Generative Processes in Music". The Psychology of Performance, Improvisation, and Composition. Oxford Science Publications (1988).

[3] Clynes, M. "Generative principles of musical thought integration of microstructure with structure". Journal For The Integrated Study Of Artificial Intelligence, Cognitive Science And Applied Epistemology 3 (1986) 185-223.

[4] Clynes, M. "Microstructural musical linguistics: composers' pulses are liked most by the best musicians". Cognition. International Journal of Cognitive Science, 55, (1995), 269-310.

[5] Davidson, J. W. and North, A. C. The Social Psychology of Music. Oxford University Press (2006).

[6] Oosten, P. van "Critical study of Sundberg's rules for expression in the performance of melodies". Contemporary Music Review, 1993, Vol.9, 267-274

[7] Poli, G. D. "Methodologies for expressiveness modelling of and for music performance". Journal Of New Music Research 33 (2004) 189-202.

[8] Repp, B. H. "Diversity and commonality in music performance: An analysis of timing microstructure in Schumann's Traumerei". Journal of the Acoustical Society of America (92).

[9] Temperly, D. The Cognition of Basic Musical Structures. The Mit Press (2004).

[10] Widmer, G. and Goebl, W. "Computational models of expressive music performance: The state of the art". Journal of New Music Research 33 (2004) 203-216.