Abstract

Research on the application of Interactive Evolutionary Computation (IEC) to the field of musical computation has been improved in recent years, marking an interesting parallel to the current trend of applying human characteristics or sensitivities to computer systems. However, past techniques developed for IEC-based composition have not necessarily proven very effective for professional use. This is due to the large difference between data representation used by IEC and authored classical music composition. To solve this difficulties, we purpose a new IEC approach to music composition based on classical music theory. In this paper, we describe an established system according to the above idea, and detail of making success of composition a piece.

1 Introduction

The most essential consideration when applying Evolutionary Computation to problem solving is the encoding that represents the problem as the gene. The efficiency of the creative evolution depends to a large degree on the representation utilized.

Also user interface and process of operation are important in Interactive EC. This is due to the problem that there are limitations to the population size and number of generation which user can deal with in IEC because of the user’s burden. The previous researches tried to solve these problems of Interactive EC are described in (Biles 1994; Takagi and Ohya 1996; Tokui and Iba 2000; Unemi 1998).

For application to IEC for the sake of composition assistance, various gene representations and user interfaces have been tried. A general review of the application of EC, especially GA and GP, to composition can be found in (Burton and Vladimirova 1999).

Traditional composers often use tree topology to represent the result of analysis for their pieces. For this reason, tree representation of a musical phrase has advantages in the sense that it can easily be understood by the traditional composers in particular. Tree topology that represents relationships between each note and chord are proposed in (Johanson and Poli 98; Tokui and Iba 2000). Applying the presented IEC techniques in past researches, e.g. user refine or define genome directly and mask a part of genome manually, also shows the advantage of tree representation. In addition, tree representation that can represent musical repetition of typical classical pieces with recursive tree topology was proposed in (Dahlstedt and Nordahl 2004), extending the tree representation of musical phrase.

However the representation of a musical phrase with tree topology has some problems. The problem is that tree topology easily becomes too complex to represent a comparatively long piece. In this case, the user - composer cannot understand the tree easily, in consequence, apllication of the IEC techniques that refine and define the genome manually becomes more difficult. Furthermore, dealing with complex and large trees degrades the performance of EC. For this reason, it was difficult to generate pieces with remarkable length with the systems that were present in past researches is difficult.

In order to solve these problems, we have constructed a new IEC system named CACIE(Computer Aided Composition using Interactive Evolution). CACIE is a system with the aim of aiding composers in the composition of traditional atonal pieces.

2 Construction of System

2.1 Concepts of System

The features of the CACIE system are as follows:

1. The user interacts with evolution any process actively.
2. The system and gene representations are based on traditional musical composition.
3. The system can generate long pieces without aggravation of convergence.
These features are realized by mechanisms, such as Multi-Field User Interface, Multiplex Structure, Genome Storage, gene representation and special genetic operations. We will mention each mechanism in the following sections.

2.2 Methods

Tree Representation of Musical Phrase  Figure 1 shows the simple example of conversion of a musical phrase into a tree-topology of the CACIE. The tree consists of terminal nodes and non-terminal nodes. Each terminal node contains a note or a musical motive. In the first example of this figure, c, d, e and f are terminal nodes. A note contains four parameters: Note Number, Amplitude, Duration and Onset-Time. Zero amplitude represents a rest note. On the other hand, in the non-terminal node, functions that merge or concatenate the notes into larger musical structures, are represented as lists of notes. In the second example of the figure, S and U are non-terminal nodes. The S function connects two nodes being as leaf, lists of notes continuously. The U function merges two nodes that will be played simultaneously. The system provides any functions that realize traditional musical structure simply in addition to S and U. Table 1 presents a partial list of functions that have been implemented. Besides, these functions are provided as libraries for user’s programming. The user can program and include new functions, new musical ideas, into the system with the combination of these functions easily.

Recursive Terminal Node  The authors designed special type of terminal node that named Recursive Terminal Node. The recursive terminal node is a simple method of representing repetition of classical music pieces. A short genome with many notes of musical significance can be constructed using this kind of the node. Before the ontogeny phase when phenotype are created, recursive terminal nodes develop the tree recursively. The recursive node replaces itself with the sub-tree of upper node. During replacement, the recursive terminal node in the new sub-tree becomes a normal terminal node. As shown in Figure 2, the expansion of a recursive terminal node is displayed.

Genetic Operation  The authors adopted two more special genetic operations as mutation, i.e., swap of sub-trees and one point crossover of linear GP. These additional special genetic operations are named Increase and Decrease. The increase mutation is a implementation of recursive terminal node, mentioned above, as mutation. Nodes that are applied to this operation will be selected randomly.

The reverse role of increase mutation is decrease mutation.

Table 1: List of part of functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>Connect two arguments continuosly</td>
<td>((S \ a \ b) = (a \ b))</td>
</tr>
<tr>
<td>U</td>
<td>Connect two arguments simultaneously</td>
<td>((U \ a \ b) = (ab))</td>
</tr>
<tr>
<td>SR</td>
<td>Make a repetition of two arguments</td>
<td>((SR^5 \ a \ b) = (a \ b \ a \ b \ a))</td>
</tr>
<tr>
<td>D</td>
<td>Apply rhythm pattern of 2nd argument to 1st argument</td>
<td>(D \ a(60,100,10) \ b(62,120,20)) = a'(60,100,20))</td>
</tr>
<tr>
<td>P</td>
<td>Apply pitch pattern of 2nd argument to 1st argument</td>
<td>(P \ a(60,100,10) \ b(62,120,20)) = a'(62,100,10))</td>
</tr>
<tr>
<td>RV</td>
<td>Reverse ordering</td>
<td>((RV \ (a \ b)) = (b \ a))</td>
</tr>
<tr>
<td>IV</td>
<td>Pitch Inversion</td>
<td>((IV \ a((60,100,10)(62,120,20)) = a'((62,100,10)(60,120,20)))</td>
</tr>
<tr>
<td>TP</td>
<td>Pitch Transposing</td>
<td>((TP + 5 \ a(60,100,10)(62,120,20)) = a'(65,100,10)(67,120,20)))</td>
</tr>
<tr>
<td>MS</td>
<td>Return a sequence of arranged notes taken from two nodes alternately as follows:</td>
<td>((MS \ (a \ b \ c \ d) = (a \ d \ b \ c \ e)))</td>
</tr>
<tr>
<td>MU</td>
<td>Return a sequence of arranged notes taken from two nodes simultaneously as follows:</td>
<td>((MU \ (a \ b \ c \ d) = ((U \ a \ c) \ (U \ b \ d)))</td>
</tr>
<tr>
<td>CAR</td>
<td>Return a sequence that contains the front X% of number of notes as follows:</td>
<td>((CAR50% \ (a \ b \ c \ d) = (a \ b)))</td>
</tr>
<tr>
<td>FILP</td>
<td>Return a sequence that contains a repetition with pitch transposing until second node pitch as follows:</td>
<td>((FILP + 2 \ (a(60,amp,dur) \ b(63,amp,dur)) = c(65,amp,dur)))</td>
</tr>
</tbody>
</table>

\[\begin{align*}
  a'(62,..) & = a'(60,..)
  b'(65,..) & = b'(63,..)
  a''(64,..) & = a''(62,..)
  b''(67,..) & = b''(65,..)
  c(65,..) & = c(65,..)
\end{align*}\]

Figure 1: Tree representation of musical phrase.
Multi-Field User Interface  We adopted the Multi-Field User Interface (Unemi 1998) in which offsprings are displayed in a separate window for the CACIE. This system does not replace elements of the population immediately. The user can compare and listen to the phenotypes of parent and offspring populations. If the user does not take to the temporary offspring, they may re-attempt the reproduction phase. In this case the user can take selected individuals from genome storage and re-injected the population, which will be discussed in more detail later on. There is assumed to be a small population, thus the process of this comparison should not be very complex.

Multiplex Structure of Composition Process  The process of composing a long piece with the CACIE is based on the Multiplex Structure, according to the model of traditional musical composition. This method is used for solving problems of bloat and complexity to the user’s advantage.

In traditional composing, short pieces such as the Invention or Lieder are typically started by deciding which notes to use for the piece. The notes are then ordered. In the next step, the composer combines the array of notes obtained in the first step and composes a “Motif”. Lastly, the pieces are composed by transforming the Motive.

The authors have adopted this compositional method to the IEC process. In the CACIE, user composes a piece step by step, not on their first attempt such as the traditional composition. This multiplex structure is realized by the implementation, where each phenotype of the results run will be exported as MIDI event lists, then the user moves up to the next step including (by many) exported MIDI event lists as a terminal node.

The user repeats each step as many times as desired to collect the materials to be used for the next step. In the next iteration, the user tries various combination of functions and terminal nodes.

Genome Storage  There is an external window in the GUI of the CACIE named Genome Storage. The genome storage is a temporary storage space by which strong intervention is realized for the user to evolution process as a part of multi-field user interface. The user can store individuals from the population in the genome storage anytime. Moreover, an important aspect of the IEC technique is that users can re-inject individuals into the population anytime; this results in the user’s active intervention in the act of evolution.

Also, the genome storage has a function that import and export individuals as a text format which includes genome tree topology and MIDI event list as phenotype. Multiplex structure, mentioned before, is realized by these functions. Typical process of each step is shown in Figure 4.

3 Experimental Results

Evolving a Melody or Piece Step  To verify the validity of multiplex structure and gene representation, we have tried to generate a long piece with the system. In this case, we adopted a technique in which the user can refine the presented genome as well as store and re-injection. The acquired pieces are available from the World Wide Web as SMF.

We have confirmed that multiplex structure works effectively to reduce the user’s burden when the user tries to generate a long piece at one time. Because of this functions, most of the results contain musical repetition from small to large level structure. Moreover, the mechanism that user can refine the presented genome helps faster convergence even in small population size.

Lastly, we tried to case of the fully compose a piano miniature with the CACIE. The result contains large level musical structure, A-B-C-D-A’-B’-E. We can show this structure in the typical traditional classical musical pieces. Furthermore,
each of the melodies to consists of the repetition of notes or small array of notes.

Full musical score of the generated piece available from our Web Site\(^2\). In addition, we asked a professional classical pianist to play piece to estimate the generated piece. The pianist had a good opinion of the piece. Performed SMF file with the MIDI piano is also available from same Web Site.

4 Discussion

4.1 Evaluation of the System

We asked any classical composers to evaluate the system, then to fill out a questionnaire in free style. We received several favorably points, also some unpopularity points.

The favorably points are same as features of the CACIE presented in section 2.1. The fact provides evidence that the system concepts and methods are successful.

On the other side, unpopular point was that referring and revising interface is not constructed with a GUI. They said that programming musical phrase and structure with the system is flexible, by including GUI programming environment, the system would be more convenient.

4.2 Problems

As mentioned in section 4.1, professional composers appreciated the concepts of this research and implementation. However, there are some problems recognized by the authors.

Processing speed of generating evaluable data from gene is very important for Musical Interactive EC. Because, waiting time for evaluation reduce user’s interactive feeling, especially the reduction is remarkable in case of application to musical creation. Therefore, developing better gene representation and improvement of implementation, such as including parallel processing are required.

As mentioned in section 4.1, Also GUI programming environment for the tree representation is most urgent task.

5 Conclusion

In this paper, we reported a research about our new system on applying Evolutionary Computing to computer aided composition system, in order that a traditional musical composer can use IEC system for their actual creation actively. Basic ideas of the presented system and gene representation are based on the traditional musical composition technique. As a result, we have succeeded to generate comparatively a long piece that include traditional musical expressions.

References


\(^2\)http://www.iba.k.u-tokyo.ac.jp/~dando/public/works/rattfylla.html

257