MELEC: Meta-Level Evolutionary Composer

Andres Calvo
Department of Computer Science, The University of Dayton
Dayton, OH 45469, U.S.A.

and

Jennifer Seitzer
Department of Computer Science, The University of Dayton
Dayton, OH 45469, U.S.A.

Abstract

Genetic algorithms (GA's) are global search mechanisms that have been applied to many disciplines including music composition. Computer system MELEC composes music using evolutionary computation on two levels: the object and the meta. At the object-level, MELEC employs GAs to compose melodic motifs and iteratively refine them through evolving generations. At the meta-level, MELEC forms the overall musical structure by concatenating the generated motifs in an order that depends on the evolutionary process. In other words, the structure of the music is determined by a geneological traversal of the algorithm's execution sequence. In this implementation, we introduce a new data structure that tracks the execution of the GA, the Genetic Algorithm Traversal Tree, and uses its traversal to define the musical structure. Moreover, we employ a Fibonacci-based fitness function to shape the melodic evolution.

Keywords: Artificial Intelligence, GATT, Genetic Algorithms, MELEC, Music Composition

1 INTRODUCTION

Genetic algorithms (GA's) emulate the mechanics of natural selection to solve a specific problem by iteratively producing new generations of candidate solutions [3]. The general algorithm of a GA system is a cycle of generate-measure-select activities. First, a GA generates a certain quantity of candidate solutions called chromosomes, the set of which are known as a generation. Then, the merit of each chromosome is assessed by a fitness function. Last, the GA selects candidate solutions based on their fitness values to form the next generation. This process is repeated as many times as necessary to achieve a near-optimal solution.

In this paper we present MELEC, a music composition system that uses evolutionary computation on two levels: the object and the meta. At the object level, we employ genetic algorithms to compose melodic motifs and iteratively refine them through evolving generations. At the meta level, we form the overall musical structure by concatenating each generation's output. That is, the structure of the music is formed by a geneological traversal of the algorithm's execution sequence. We introduce a new data structure, The Genetic Algorithm Traversal Tree (GATT), that tracks the execution of the GA and uses its traversal to define the musical structure. Moreover, we employ a Fibonacci fitness evaluation to shape the melodic evolution.

2 BACKGROUND

Generating musical phrases with GA's has been studied and implemented in several ways. Most commonly, a GA is used to produce several musical phrases that must somehow be concatenated or combined to produce a final composition. The tree in Figure 1 depicts many musical composition systems that have been created using GA's. This taxonomy, established in [3], is based on the type of fitness functions employed. We provide a representative survey of these systems.

- *Deterministic* evaluation methods utilize mathematical functions to calculate a musical event's fitness. For instance, The autonomous evolutionary music composer (AEMC) generates several motifs by using a GA with a fitness function which favors successive notes that have a difference of less than 7 half steps [4]. AEMC then combines the resulting motifs and their transpositions using a fitness function which considers the ratios of notes witin the melody. This fitness function favors an arbitrarily chosen note ratio of 60% tonal centers, 35% color notes, and 5% chromatic notes. These ratio values were arbitrarily selected and serve as a starting point to begin further research [4].

  Another deterministic compositional system is pre-

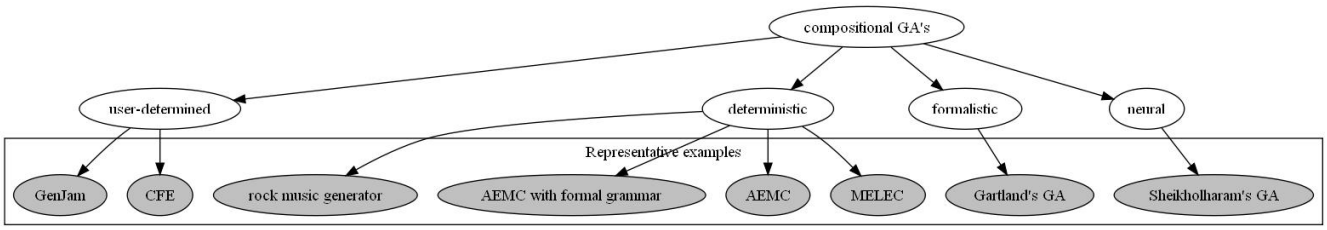


Figure 1: Tree depicting a compositional compositional GA's classification based on fitness function

sented in [5] and uses a GA with a formal grammar evaluation function. A formal grammar is 'a collection of (...) descriptive or prescriptive rules for analyzing or generating sequences of symbols' and gives the fitness function abilities ranging from checking for tonality to encouraging certain fundamentals of music theory [5].

A last deterministic system example is described in [7], where Oliwa considers a variety of parameters to generate music for several instruments commonly used in rock music: guitar, drums, and piano/organ. Oliwa uses separate fitness functions to customize each instrument's role within the music. For instance, Oliwa uses a fitness function that favors drum patterns that are repeated up to four times and are suceeded by a filling pattern.

- *Formalistic* rules encode stylistic features from previous music as the set of rules which form the fitness function. For example, Gartland uses a GA with a fitness function that favors music which contains common features with a predefined piece of sample music.
- *Neural* fitness functions use neural networks to favor musical events which are similar to desired events incorporated in its training data. For instance, Sheikholharam and Teshnehlab use a GA to compose music and define its fitness function by using a recurrent neural network to determine desired characteristics from existing music [8].
- *User-determined* evaluation methods require a user to quantify a musical event's fitness function. The user's subjective input stipulates the desired musical style. For instance, the CFE framework allows a user to hear and rate melodies to determine the value of their fitness functions [2]. The user's input is fed back into the GA in order to continue the musical evolution. Another user-determined system is GenJam, which uses a GA to produce jazz music improvisation on a single instrument [5]. GenJam allows a user to quantify a musical phrase's fitness and reduces the number of phrases a user must evaluate by removing measures that are characterized as unmusical through the use of a neural network.

The system presented in this paper is called the meta-level evolutionary composer (MELEC) and utilizes a GA with a deterministic Fibonacci-based fitness function to generate motifs. Unlike other deterministic compositional GA's, MELECs fitness function favors motifs in which the number of half-steps between adjacent notes falls in the Fibonacci sequence. Furthermore, MELEC utilizes the meta-level information behind the GAs evolution to concatenate motifs from different generations to form a melody. In contrast to traditional GA's, MELEC stores all formed generations and incorporates the evolutionary processs meta-level information into the formation of the final melodys structure.

3 THE MELEC SYSTEM

The system generates motifs through the use of a GA and combines them to form a melody. The meta-level information from the evolutionary process determines the order in which the motifs are combined. MELEC assumes that every motif consists of 16 eighth notes. Notes are represented through the Musical Instruments Digital Interface (MIDI) protocol since it conveniently characterizes musical parameters such as pitch and duration. Furthermore, most media players support MIDI for easy playback. A MIDI pitch is represented by a number between 0 and 127 where middle C is represented by 60 and where two consecutive numbers are separated by a half-step. MELEC generates melodies in which all notes are constrained to the two-octave range of pitches between 48 and 71. MELEC divides the compositional process into two phases. The first phase generates several melodic motifs and stores each one as a node in the GATT data structure. The second phase traverses the GATT and concatenates each nodes motif to form an overall melody.

3.1 Phase 1: Generating motifs with a GA

The GA contains parameters that define the number of motifs **N** in each generation, as well as the total number of generations **G**. The initial generation consists of **N** randomly generated motifs. Algorithm 1 shows the steps by which MELEC generates motifs.

The *Genetic Algorithm Traversal Tree* (GATT) is a data structure which stores the motifs from every generation and keeps track of every child node's parents. The GATT

Input: N, G
Output: GATT containing N*G motifs
for *G generations* **do**
  generate N chromosomes (melodies) and attach to the GATT in a breadth-first manner;
  run each of the chromosomes through the fitness function;
  select parents for the next generation using proportionate fitness selection;
end

Algorithm 1: Building the GATT Structure while evolving the melodic stream.

therefore stores the generated motifs' geneological history. In typical GA systems, only the final generation of candidate solutions is considered relevant while the rest are discarded. In contrast, MELEC stores and uses motifs from all generations. Thus, the GATT differentiates MELEC from other composition systems by allowing its output to be a representation of the evolutionary *process* as opposed to an optimized result.

MELEC generates new (motif) chromosomes in pairs. To create two chromosomes in the next generation, MELEC selects two motifs from the current generation and performs the standard GA operations of single-point cross-over and mutation. A fitness function then evaluates each motif to assess its merit and to determine its probability ofgetting selected as a parent for the next generation. The chromosome is then encapsulated in a vertex and inserted into the GATT as a child node of its parents. Although the generation size of MELEC is a user-specified parameter, in our tests, we chose generation sizes of 6 and 16.

3.2 Fitness function

MELEC's fitness function evaluates the intervalic distance between adjacent notes. If the interval (measured in half steps) falls in the *Fibonacci sequence*, the fitness value is increased by 2000, if not, it is ignored.

The Fibonacci sequence is defined by the recurrence relation:

$$F_0 = 0$$
$$F_1 = 1$$
$$F_N = F_{N-1} + F_{N-2}$$

We chose to incorporate the Fibonacci Sequence as an integral part of our fitness function because of its frequent appearance in music [6]. The sequence appears as a melodic underlying primitive as well as structural foundation in many pieces of music. MELEC's fitness function is outlined in Algorithm 2.

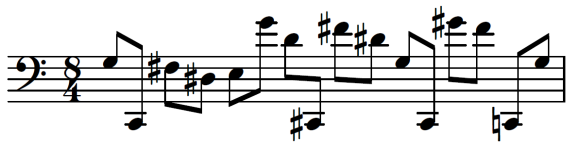
In the next subsection, we illustrate the first phase of composition with an in-depth example of MELEC's implementation of the create-evaluate-select GA cycle.

Input: Motif M
Output: M's fitness value
Fitness = 0;
for $i = 0; i < 14; ++i$ **do**
  if $|M[i+1] - M[i]| \in F$ **then**
    Fitness = Fitness + 2000;
  end
end

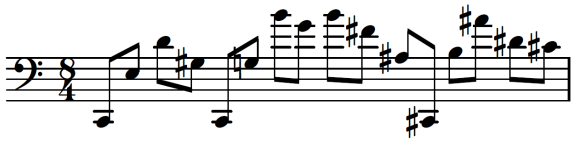
Algorithm 2: The Fibonacci fitness function

3.3 Example motif generation

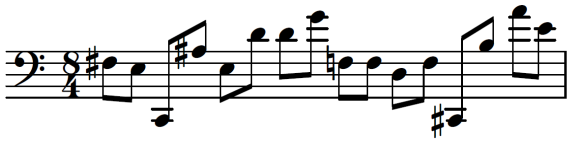
Consider the following *first generation* of randomly generated motifs as shown in Figure 2. For this example, we assume that a generation consists of three motifs.



(a) Motif 1 of Generation 1.



(b) Motif 2 of Generation 1.



(c) Motif 3 of Generation 1

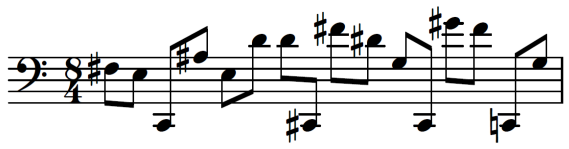
Figure 2: The resulting first generation.

A *Fibonacci interval* is defined as an interval that belongs to the Fibonacci sequence when measured in half-steps. To calculate the fitness function for the first measure (i.e., Figure 2a), we count the number of Fibonacci intervals it contains and multiply by 2000. The first measure in the generation has eight Fibonacci intervals: between notes 3 and 4, notes 4 and 5, notes 6 and 7, notes 7 and 8, notes 9 and 10, notes 10 and 11, notes 11 and 12, and notes 13 and 14. Therefore, the first measure has a fitness value of 16000. The fitness values for motifs 2, and 3 are 12000 and 14000, respectively.

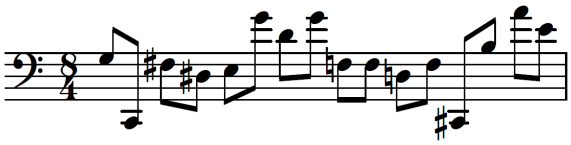
The GA then selects two parent nodes through fitness proportionate selection. For this example, we assume that the first (Figure 2a) and the third (Figure 2b) motif were selected.

Two children are formed by recombining and mutating the selected parents. Single-point crossover about a random point (the seventh note in this example) obtains the motifs shown in Figure 3.

We've arbitrarily chosen the probability of each note in a



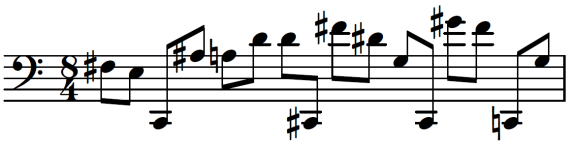
(a) First motif resulting from crossover.



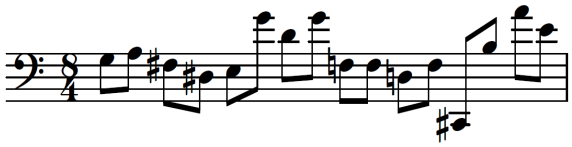
(b) Second motif resulting from crossover.

Figure 3: Recombining the selected parent motifs.

motif mutating to be 10%. after the crossover has occurred. Mutating a note replaces its pitch by a random value within the valid range. The resulting motifs, shown in Figure 4, are the first two candidate solutions for the next generation.



(a) First motif.



(b) Second motif.

Figure 4: The second generation.

The child motifs in generation two are then related to their parents in the GATT as illustrated by Figure 5.

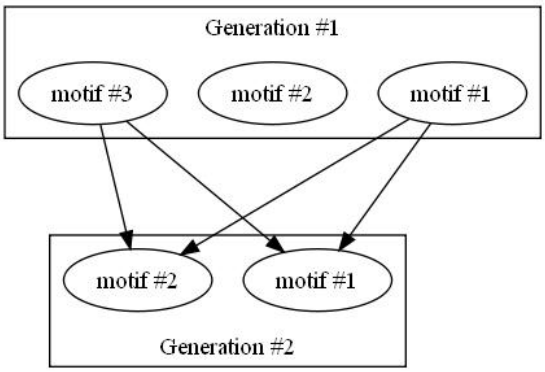


Figure 5: Resulting GATT structure for the presented example.

3.4 Phase 2: Traversing the GATT

The GA generates a large number of motifs which collectively lack the musical arrangement required to form the melody line of a complete piece of music. The meta-level information established as the composition's GATT is used to remedy this problem. Concatenating the motifs based on a GATT traversal results in a composition which reflects the evolution of the random motifs into more 'listenable patterns'. The use of contiguous generations in the melody stream provide a sense of similarity and repetition for the listener. The similarities between a parent and its children allow the composition to retain a characteristic sound which stimulates a sense of familiarity in listeners. The meta-level information permits the graph traversal to take the listener through the process of the evolution and refinement of the music, thereby providing both content and structure for the music.

Section 3.3 illustrates how MELEC generates motifs and attaches them to the GATT. At the end of phase 1, MELEC has completed generating motifs and creating the GATT. Phase 2 creates the musical composition by concatenating motifs in an order dictated by various traversals of the GATT. Schemes such as as pre-, in-, and post-order GATT traversals can be used to form the melody line of the overall piece. Algorithm 3 describes this traversal process.

Input: The GATT
Output: A melody
1 Select the number of motifs which will form the melody;
2 Using one of the traversal schemes, visit the nodes in the GATT and concatenated into a melodic stream until the desired motif length is reached;

Algorithm 3: Traversing the GATT structure to form a melody.

4 SOFTWARE ARCHITECTURE

The MELEC system was programmed in C# following a modular object-oriented approach. MELEC's software architecture is depicted in Figure 6. The *Note* class abstracts a musical note and stores its pitch, duration, and velocity. The *Measure* class abstracts motifs formed by instantiating 16 *Note* classes. The *Measure* class provides methods for playing the entire motif and writing the measure to a MIDI file. The MidiFile class contains the methods necessary to specify a MIDI file's tempo and time signature, and ensure that it is formatted according to the MIDI standard.

The GA was encapsulated in the *GeneticMidi* class, which calls methods from the *Fitness*, *Crossover*, and *Mutation* classes. The *GeneticMidi* class generates the final performance and stores it as a MIDI file. The MIDI file can then be played with Windows Media Player.

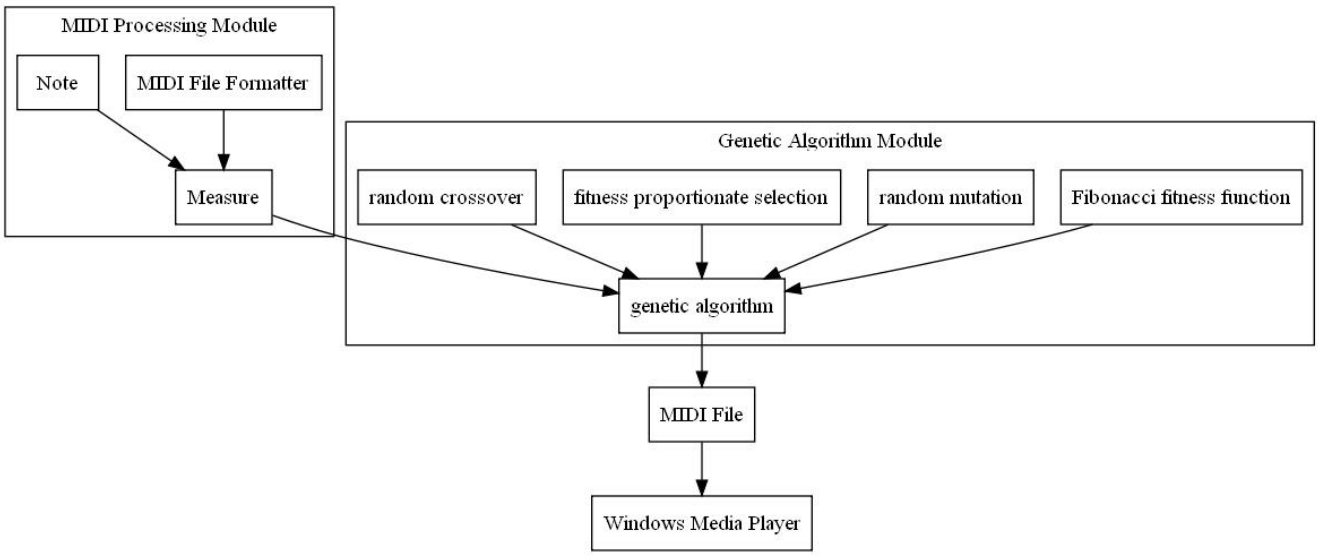


Figure 6: Diagram depicting MELEC's class structure

5 MUSICAL QUALITY

Generating musical phrases that are pleasant to listen to can be an arduous task. Work in psychology and music [1] indicates that several factors come into play. Is the next melody predictable from the first? Is the sequence of notes pleasant? Does the structure of the music defy predictability? Is there a pattern to rhythmic modulations? Several scientist-musicians have investigated the brain as it listens to 'good' music. In [6], Levitin presents the idea that the human brain is hard-wired to understand music, possibly as a result of evolution, since humans listened to music (drumming) before speech. He claims that we are invoking very ancient and primitive parts of the brain while listening to music. Moreover, by rewarding melodies that exhibit traces and intervals in the Fibonacci sequence, we appeal to an aesthetic proclivity that seems to be ubiquitous in the Arts: Fibonacci numbers are omnipresent in architecture, literature, and music.

Forming a melody through a GATT traversal imposes a musical structure by placing similar motifs together. The generated melodies impose a sense of familiarity in listeners because the musical arrangment reflects the GA's execution.The GATT enables us to provide melodies with structure while allowing the GA to provide creativity at the motif level.

6 CONCLUSIONS

MELEC approaches musical composition by capturing the process of a GA's evolution as opposed to its final generation of optimized candidate solutions. By using the Fibonacci fitness fuction, MELEC generates musical motifs that evolve from an initial generation of random notes to a final generation of notes where Fibonacci intervals are predominant. Through the use of the GATT, we are able to arrange the motifs in an order that reflects their evolution. Traversing the GATT generates melodies formed by motifs that increase and decrease their number of Fibonacci intervals in an ordered manner. The traversal imposes a sense of familiarity and predictability in the generated melodies which are perceived as musical.

7 FUTURE WORK

As we continue working with music composition with MELEC, we expect to experiment with different traversals of the GATT. We intend to try pre-order, post-order, breadth-first, depth-first, and a best-first A* approach to musical structure definition. We also anticipate experimenting with generation choice in forming the GATT. For example, how does the music sound when only *prime numbered* generations are included in the piece. We also expect to augment the implementation of MELEC to provide musical compositions with fully scored harmonies and instrumentation specifications.

References

[1] BALL, P. *The Music Instinct: How Music Works and Why We Can't do Without it*. Random House, London, 2010.

[2] CHEN, Y. Interactive music composition with the cfe framework. *SIGEVOlution 2*, 1 (2007), 9–16.

[3] JRVELINEN, H. Algorithmic musical composition, 2000.

[4] KHALIFA, Y., AND AL-MOURAD, M. B. Autonomous evolutionary music composer. In *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation* (New York, NY, USA, 2006), ACM, pp. 1873–1874.

[5] KHALIFA, Y. M. A., KHAN, B. K., BEGOVIC, J., WISDOM, A., AND WHEELER, A. M. Evolutionary music composer integrating formal grammar. In *GECCO '07: Proceedings of the 2007 GECCO conference companion on Genetic and evolutionary computation* (2007), pp. 2519–2526.

[6] LEVITIN, D. *This Is Your Brain on Music: The Science of a Human Obsession*. Dutton, Boston, 2007.

[7] OLIWA, T. M. Genetic algorithms and the abc music notation language for rock music composition. In *GECCO '08: Proceedings of the 10th annual conference on Genetic and evolutionary computation* (2008), pp. 1603–1610.

[8] SHEIKHOLHARAM, P., AND TESHNEHLAB, M. Music composition using combination of genetic algorithms and recurrent neural networks. In *HIS '08: Proceedings of the 2008 8th International Conference on Hybrid Intelligent Systems* (Washington, DC, USA, 2008), IEEE Computer Society, pp. 350–355.