Evolutionary Music Composer integrating Formal Grammar

Yaser M.A. Khalifa
Electrical and Computer Department
State University of New York
New Paltz, NY 12561 USA
257-3764, 845
khalifay@newpaltz.edu

Badar K Khan
Electrical and Computer Department
State University of New York
New Paltz, NY 12561 USA
338-4705, 845
khan38@gmail.com

Jasmin Begovic
Electrical and Computer Department
State University of New York
New Paltz, NY 12561 USA
546-6856, 845
begovi41@newpaltz.edu

Airrion Wisdom
Electrical and Computer Department
State University of New York
New Paltz, NY 12561 USA
257-3764, 845
wisdom76@newpaltz.edu

Andrew Maxymillian Wheeler
Electrical and Computer Department
State University of New York
New Paltz, NY 12561 USA
338-4705, 845
wheele49@newpaltz.edu

Abstract**—**In this paper, an autonomous music composition tool is developed using Genetic Algorithms. The production is enhanced by integrating formal grammar rules. A formal grammar is a collection of either or both descriptive or prescriptive rules for analyzing or generating sequences of symbols. In music, these symbols are musical parameters such as notes and their attributes. The composition is conducted in two Stages. The first Stage generates and identifies musically sound patterns (motifs). In the second Stage, methods to combine different generated motifs and their transpositions are applied. These combinations are evaluated and as a result, musically fit phrases are generated. Four musical phrases are generated at the end of each program run. The generated music pieces will be translated into Guido Music Notation (GMN) and have alternate representation in Musical Instrument Digital Interface (MIDI). The Autonomous Evolutionary Music Composer (AEMC) was able to create *interesting* pieces of music that were both innovative and musically sound.

Categories and Subject Descriptors
I.2.0 [General]

General Terms
Experimentation, Human Factors.

Keywords
Music, Formal Grammar, Genetic Algorithms.

I. INTRODUCTION

A number of evolutionary based algorithms have recently been documented in literature. The concept of algorithmic composition has long been attempted in the past, however, due to the nature of music as a creative activity, there is still a need for further work in this area. In [1], Gartland-Johnes and Colpey provide an excellent review of the application of Genetic Algorithms in musical composition. Miranda, in [2] discusses different approaches to using evolutionary computation in music. However, most systems listed in literature need a tutor, or an external evaluator. In [3] Both David Cope and Robert Rowe designed systems that created music by analysis of other music. While Cope's system took input from classical compositions, Rowe looked at live input from a performer. Rowe drew a distinction between transformative and generative music composition. While his ``Cypher" system contained elements of both, it was primarily a transformative one -- listening to the input from the user, pushing the input through a series of transformations, and then outputting something derivative, although not necessarily reminiscent.

Biles and colleagues have used a neural network critic with the GenJam system [4], and to unclog the fitness bottleneck caused by presenting a user with too many musical examples to evaluate, these researchers hoped a neural network critic could at least filter out measures that were clearly unmusical before they reached the user. Results reported wre not encouraging since population quickly finds a loophole in the fitness function and presents cheating solutions that will have such a fitness advantage over other members of the population that they will rapidly take over, killing off any other alternative approaches. This is not such a problem with human critics (as noted by Biles et. Al. in [4]), because their selection criteria can change over time to search for new aspects in creators and thus avoid stagnation.

The development of autonomous unsupervised music composers is therefore still very limited, but yet has lots of potential. In addition to that, the concept of using pattern extraction techniques to extract primary patterns, or motives, in established pieces of music has not been extensively explored in the literature. This is somewhat surprising, since composers have made use of motives for composition for centuries. The problem of composing music based on a library of motives is, however, near or perhaps slightly beyond the frontier of current capabilities of artificial Intelligence (AI) technology. Thus, this area of research spearheads a new direction in automated composition.

The work presented in this paper is an attempt in that direction. It presents an autonomous music composition system. The system composes musical pieces based on a library of evolving motifs. The critique of the generated pieces is based on three evaluation functions: intervals, ratios, and formal grammars that each describes an aspect of the musical notes and/or system as explained in the following sections.

II. Genetic Algorithms Implementation

GA are a stochastic combinatorial optimization technique [8]. It is based on the evolutionary improvement in a population using selection and reproduction, based on fitness that is found in nature. The GA operates on a population of chromosomes, where each chromosome consists of a number of genes, and each gene represents one of the parameters to be optimized. The composition of music is performed in two Stages. In Stage I, a set of motifs is generated. In Stage II, motifs and their transpositions are combined to form two music phrases, A and B. At the end of Stage II, phrase $A^{\#}$ is generated by sharing each note of the phrase. At the end, a combination of $ABA^{\#}A$ is produced, which is one of the common combinations in music composition theory.

III. Music Background

In this section, some basic fundamentals of music composition are given. Because the piano has a good visual explanation of music, it will be used for illustration, however these concepts can be transposed to any musical instrument including the human voice.

We begin by analyzing the most basic set of notes called the C major scale, which consists entirely of all the white notes. We will dissect what major scales are, how they composed and further our discussion to how they form what are called chords, or simultaneously depressed single notes.

Music regardless of the instrument has a maximum 12 different distinct pitches or tones which are called keys. A pitch is simply a frequency of sound; within these pitches a multitude of combinations can be formed to produce "music". However, how can we be assured that a specific combination will be musically pleasing to the ear? Of course the term "musically pleasing" is subjective to the listener, but there must be some fundamental principle underlying the organization of the combination in question.

There is an *interval* that exists between to consecutive pitches, the term musical interval refers to a *step* up or down in musical pitch. This is determined by the ratios of the frequencies involved. "…an octave is a music interval defined by the ratio 2:1 regardless of the starting frequency. From 100 Hz to 200 Hz is an octave, as is the interval from 2000 Hz to 4000 Hz." In music we refer to the interval between two consecutive notes as a *half step*, with two consecutive half steps becoming a *whole step*. This convention is the building block of our major scale.

A scale is a set of musical notes that provides the blueprint of our musical piece. Because our starting point is the musical note C, this major scale will be entitled as such. The major scale consists of a specific sequence of whole steps of and half steps, that being W W H W W W H, where W is a whole step and H is a half step. A typical musical convention is to number the different notes of the scale corresponding to their sequential order, usually called *roots*. Using the sequence of the major scale, our C major scale consists of the notes C D E F G A B, returning to note C completing what is known as an *octave*, or a consecutive sequence of 8 major scale notes. Numeric values are now assigned where C corresponds to value 1, D would be 2, E being 3 and so on. The next C in terms of octaves would restart the count therefore the last value or root would be 7 corresponding to note B.

We build on these scales by combining selected roots simultaneously to form what are known as *chords*. Chords can be any collection of notes, this leads to almost endless possibilities in music, however for our purposes we implement the *C major chord* and use its sequence of notes. A major chord consists of the 1st, 3rd, and 5th root of the major scale, this would mean that we utilize notes C E and G.

IV. Stage I

In Stage I, motifs are generated. A table of the 16 best motifs is constructed that is used in Stage II. These motifs will be used both in their current, and transposed locations to generate musical phrases in Stage II. Fig .1 shows the chromosome structure in Stage I. Each chromosome will contain 16 genes, allowing a maximum of 16 notes per motif. Each motif is limited to a four-quarter-note duration.

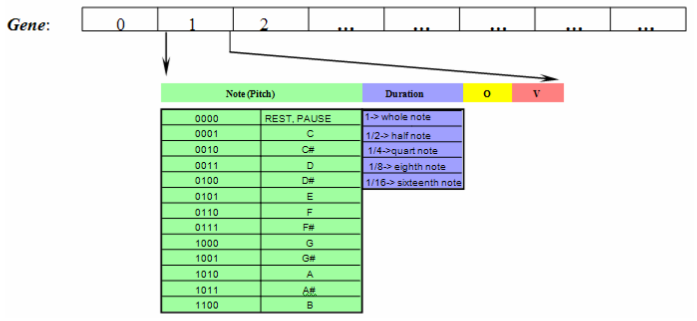


Fig .1. Chromosome and Gene Structure for Stage I

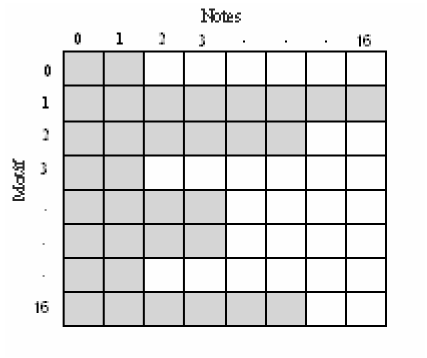


Fig .2. Motif Look-up Table Generated in Stage I

At the end of Stage I, a table of the top 16 motifs is constructed (Fig .2). Each row in this look-up table represents a motif. The columns represent the different notes in the motif. Although all motifs generated are one whole note in duration, they could be composed of either one, two, four, six, or eight notes. However, single note motifs are highly discouraged.

V. Stage I Evaluation Functions

A. *Formal Grammar Evaluation Function*

As previously mentioned the C major chord consists of the 1st 3rd and 5th root of the scale. This will correspond to values 1, 5 and 8. We can also do an *inversion* of the major chord by using the same notes C E G, this time however starting at note E leaving the inversion to be E G C. If we consider the next octave up and assign the value 13 to the repeating C, the inversion E G C will correspond to values 5, 8, and 13.

These two versions of the major chords give us two production rules of which we are assured will be musically pleasing. The production rules will take the difference of the values assigning a good fitness if the production rule is met. Our first production rule will be the difference or skip of 4 and 3 (5 - 1 = 4 and 8 - 5 = 3), describing our first version of the major chord C E G. The second rule will be the difference or skip of 3 and 5 (8 – 5 = 3 and 13 – 8 = 5), describing the inversion of the major chord E G C.

Our formal grammar can be extended from the ability to check for tonality to the ability to encourage certain fundamentals of music theory. One of the most common and important rules in music theory is the notion of the chord progression. A chord progression is simply a series of chords that are played in a particular order. The various combinations of these progressions are the basis for music we hear today. The most frequently used progressions rely on the first, fourth, and fifth degrees of the major scale. Scale degrees, are the usual nomenclature practiced when relating the name of a note with its corresponding order in which it falls on the diatonic scale. For example the diatonic major scale of C, would begin at C continuing to D, E, F G, A, and B. The scale degree of C would be the 1st, D would be the 2nd, C the 3rd, and so on. These degrees represent only a root tone; subsequent chords would be used in conjunction with that root note and other notes derived from that root to form a full musical chord. Again, we will only utilize the root and not the corresponding chord.

One of the most common progressions, primarily used in Jazz, is the II-V-I. Observing the diatonic scale of C major we see that if we start at the II, which is note D, and descend in fifths we will end at G, which is the V, and continuing in fifths we end at C which is the I. This pattern in music theory is known as the *circle of fifths*, which describes the relationships among the 12 chromatically distinct keys of music.

Another popular progression used in music is the VII-III-VI, again following the same pattern as the previously mentioned progression, which descends in fifths. In this particular progression we notice that if we end on the VI we can then move to the II if we descend following the circle of fifths theory. The II will then be able transition to the V and finally resolve at I. We now have formed generally pleasing "music" by combining the 2 progressions, namely VII-III-VI-II-V-I. This represents a pattern which can be used in producing new production rules.

Dissecting the above pattern we notice that each time we descend from a root we subtract 4 diatonic tones, and when we ascend from a root we add 3 diatonic tones. We can apply this 3-4 coupling technique to all roots within the same musical key and find that we will have musically fit pieces. Since we have chosen the key of C, our production rules are simplified due to the lack of sharps and/or flats in diatonic C major. A formal definition of the grammar we chose follows:

We define a Context Free Grammar (CFG) thusly: $G = \{N, \Sigma, P, S\}$ where N is the set of non-terminal symbols our grammar accepts, Σ is the set of terminal symbols our grammar accepts, P is a list of production rules in Chomsky Normal Form(CNF), and S is an element of N that represents the start production.

The contents of these sets are:
N: {S, A, B, C, D, E, F, G}
Σ: {a, b, c, d, e, f, g, ε}, where ε is the empty string

P: {
S → beaA | cfbB | dgcC | eadD | fbeE | gcfF | adgG
A → cfbB | dgcC | eadD | fbeE | gcfF | adgG
B → beaA | dgcC | eadD | fbeE | gcfF | adgG
C → beaA | cfbB | eadD | fbeE | gcfF | adgG | ε
D → beaA | cfbB | dgcC | fbeE | gcfF | adgG
E → beaA | cfbB | dgcC | eadD | gcfF | adgG
F → beaA | cfbB | dgcC | eadD | fbeE | adgG
G → beaA | cfbB | dgcC | eadD | fbeE | gcfF
}

This CFG allows for any number of three note tuples, with no particular note tuple repeating. For example, beabea would be rejected, but beafbebea would be accepted. Every three note tuple follows the same rule: the root note, a note three tones up in the diatonic C scale, and a final note four tones down from the previous note in the diatonic C scale. A given motif will be evaluated note by note according to this grammar, and any motif that cannot be expressed with our CFG will be rejected and given a low fitness value for the next generation, in order to encourage these fundamental progressions in our sound.

B. *Intervals Evaluation Function*

Within a melody line there are acceptable and unacceptable jumps between notes. Any jump between two successive notes can be measured as a positive or negative slope. Certain slopes are acceptable, while others are not. The following types of slopes are adopted:

Step: a difference of 1 or 2 half steps. This is an acceptable transition.
Skip: a difference of 3 or 4 half steps. This is an acceptable transition.
Acceptable Leap: a difference of 5, 6, or 7 half steps. This transition must be resolved properly with a third note, i.e. the third note is a step or a skip from the second note.
Unacceptable Leap: a difference greater than 7 half steps. This is unacceptable.

As observed from the information above, leaps can be unacceptable in music theory. We model this in GA using penalties within the interval fitness function.
Certain resolutions between notes are pleasant to hear, but are not necessary for a "good" melody. These resolutions therefore receive a bonus. Dealing with steps in the chromatic scale, we can define these bonus resolutions as the 12-to-13 and the 6-to-5 resolutions. The 12-to-13 is a much stronger resolution, and therefore receives a larger weight. It was experimentally suggested that the 12-to-13 resolution have double the bonus of the 6-to-5 one, and that the bonus does not exceed 10% of the total fitness. Thus the bonuses are calculated by dividing the number of occurrences of each of the two bonus resolutions by the number of allowed resolutions (15 resolutions among 16 different possible note selections), see equations (1) and (2).

$$\text{12-to-13 bonus} = (\#\text{occurances}/15) * 0.34 \qquad (1)$$

$$\text{6-to-5 bonus} = (\#\text{occurances}/15) * 0.34 \qquad (2)$$

The total interval fitness:

$$\text{Interval Fitness} = \frac{1}{total\_error(1 - total\_bonus)} \qquad (3)$$

VI. STAGE II

In Stage II, motifs from the look-up table constructed in Stage I are combined to form two phrases, A and B. Each phrase is eight measures, and each measure is a four quarter-note duration motif, Fig 3.

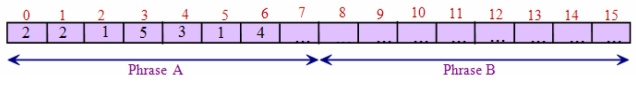


Fig .3. Chromosome Structure for Stage II

VII. STAGE II EVALUATION FUNCTIONS

In Stage II, two evaluation functions are implemented: intervals, and ratio. The intervals evaluation function described in the previous section is used to evaluate interval relationships between connecting notes among motifs, i.e. between the last note in a motif and the first note in the following motif. The same rules, as described above in Stage I, are used in Stage II. Other evaluation functions are described below.

A. *Ratios Evaluation Function*

The basic idea for the ratios section of the fitness function is that a good melody contains a specific ideal ratio of notes, and any deviation from that ideal results in a penalty. There are three categories of notes; the Tonal Centers that make up the chords within a key, the Color Notes which are the remaining notes within a key, and Chromatic Notes which are all notes outside a key. Each type of note is given a different weight based on how much a deviation in that portion of the ratio would affect sound quality. The ideal ratios sought were: Tonal Centers make up 60% of the melody; Color Notes make up 35% of the melody; and Chromatic Notes make up 5% of the melody. Although these ratios choices could be quite controversial, they are a starting point. Ongoing research is looking into making these ratios editable by the user, or music style dependent.

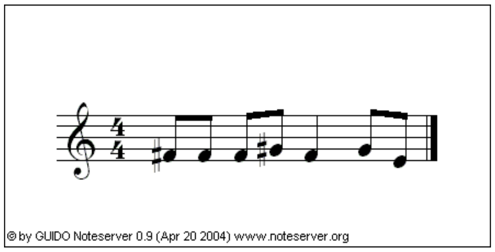
VIII. Results

A. Analysis of Motif Selection

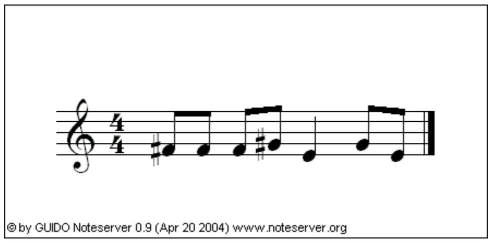
The four motifs in Fig 4 all resulted from a single running of the program. They were handpicked from the final 16 motifs selected by the program as the most fit. It can be observed that each motif has an identical rhythm consisting of four eighth-notes, one quarter-note, and two more eighth notes. Summing the durations of the notes yields the correct four quarter-note duration indicated by the time signature $\begin{pmatrix} 4 \\ 4 \end{pmatrix}$ at the beginning of each motif.

Using the intervals evaluation algorithm as a reference, we can see why these motifs were chosen to be the elite of the population. Examining motif *a*, the first three notes are all $F^{\#}$'s, indicating that no penalty will be assigned (a step size of 0). The next note is a $G^{\#}$ (2 half-steps away from $F^{\#}$). This transition is classified as a step and no penalty is assigned. The following notes are $F^{\#}$, $G^{\#}$, and E (a difference of 2, 2, and 3 half-steps, respectively). These transitions are also acceptable; therefore the intervals evaluation function would not assign any penalty to the motif. When zero error is assigned to a motif, a high fitness value will result. Similar analysis of motifs *b*, *c*, and *d* yield the same result.

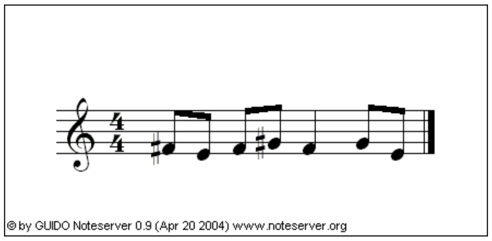
So what is the musical difference between the motifs? Since the notes in each motif are slightly different, the musical 'feel' of each motif will vary. Compare motifs *a* and *d* for example. Motif *a* contains four $F^{\#}$'s. They are arranged in such a way that the first beat and a half of the measure are all $F^{\#}$'s, and also the 3rd downbeat (the quarter-note). This repeatedly drives the sound of the $F^{\#}$ into the listener, resulting in an unconscious comparison of this note to every other note in the measure. This in turn will make dissonant notes sound more dissonant, and resolving notes sound more resolved. In the case of motif *d*, the $F^{\#}$'s are arranged in a manner that accents the steady background rhythm of the measure (the repetitive rhythm that your foot taps to when you listen to music). This does not accent the sound of the $F^{\#}$ as much, but rather accents the other rhythms of the measure that occur between the $F^{\#}$'s. A more 'primal' feel will result, as opposed to the more 'melodic' feel of motif *a*.



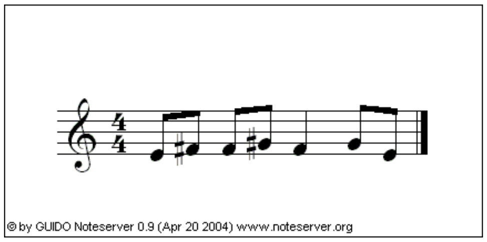
(a)



(b)



(c)



(d)

Fig .4. Sample Motif Generated in Stage I of the Evolutionary Music Composer

For the Formal Grammar evaluation function, the main musical objective is to implement the 3-4 coupling By analyzing the sequence of notes in the musical piece generated in Figure 5, it is seen that a direct correlation to our Formal Grammar production rule one. The skip from the B to D is a skip of 3 and C to E is a skip of 4 which meets our first production rule. We do see a skip of 2 that being D to C, a skip of 0 in C to C, and a skip of 1 C to B, but we must consider that formal grammar is not our only fitness criteria in the autonomous music composer.rule as previously explained in the musical theory.

A second example is shown in Figure 6, where the piece incorporates almost all the rules and musical parameters that the program has set, however let us focus on how the 3-4 coupling rule is represented. Taking a look at Figure 1 at the end of the second measure, we notice a 16th note E. We know that on the C major scale, speaking numerically, E would be represented by a 3. This would mean that to move 3 major tones up would leave us at the 6 which is B. From B in accordance with the 3-4 coupling rule, we should then descend 4 major tones, completing the rule at the 2 or note D. We see a direct implementation of this sequence twice in the entire piece. The third example is in Figure 7 and is seen at the end of the second to last measure, where the sequence is characterized by the same exact notes as explained in the first instance of the 3-4 coupling rule.

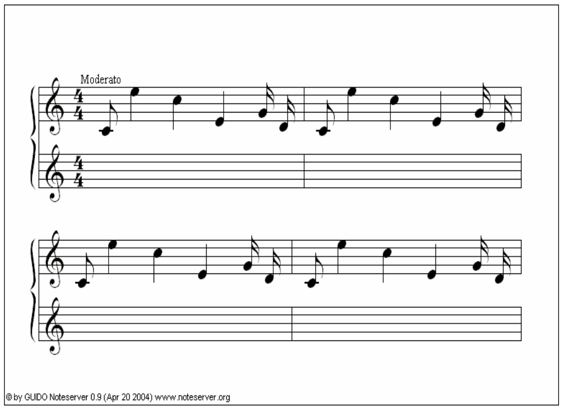


Fig .5. Sample Motifs Generated in Stage I of the Evolutionary Music Composer

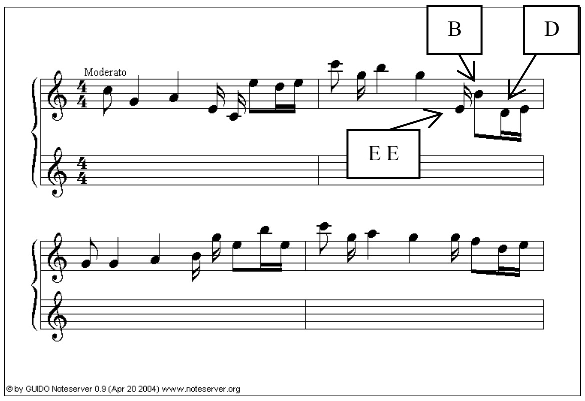


Fig .6. Sample motifs generated in Stage II of the Evolutionary Music Composer

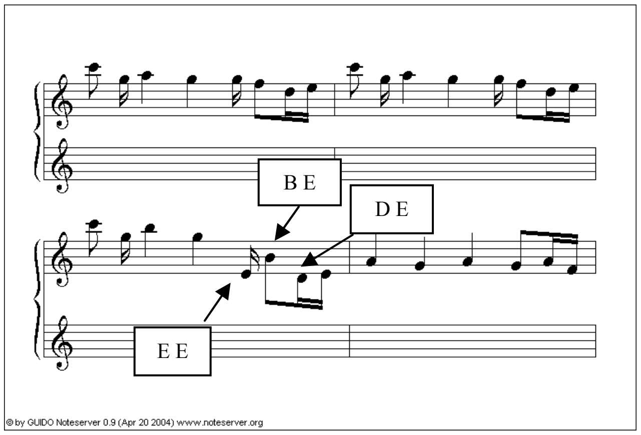


Fig .6. Sample motifs generated in Stage II of the Evolutionary Music Composer

Our second production rule is visibly met in Figure 6. The note sequence seen above is C E C E G D and repeat. The skip from E to G is a skip of 3 and G to D is a skip of 5 satisfying our second production rule. We can in this Figure however that we combine both production rules, because the skip from C to E is a skip of 4, therefore this musical piece followed our Formal Grammar rules entirely!

DISCUSSION AND FUTURE WORK

New techniques in evaluating combinations of motives are needed. The evaluation of motive combination should take into consideration the overall musical piece rather than the note transition resolutions of the first and last notes in the motif only. One approach that will be further investigated is the application of formal grammars. In a multi-objective optimization problem such as music composition, different evaluation functions are applied and contribute to the fitness measure of a generated piece. The main functions that have been implemented are intervals, and ratios. They have been equally considered in evaluating the evolutionary generated music so far. Different weighing methods for various evaluation functions is expected to effect the quality of the resulting music. These could also be affected by types of music sought, e.g. classical, Jazz, Blues, etc. In Stage II of the project, methods that use weighted combinations of different fitness functions, or composition rules, will be explored.

REFERENCES

[1] Gartland-Jones, A. Copley, P.: What Aspects of Musical Creativity are Sympathetic to Evolutionary Modeling, Contemporary Music Review Special Issue: Evolutionary Models of Music, Vol. 22, No. 3, 2003, pages 43-55.

[2] Burton, A.R. and Vladimirova, T.: Generation of Musical Sequences with Genetic Techniques, Computer Music Journal, Vol. 23, No. 4, 1999, pp 59-73.

[3] Miranda, E.R.: At the Crossroads of Evolutionary Computation and Music: Self-Programming Synthesizers, Swarm Orchestra and Origins of Melody, Evolutionary Computation, Vol. 12, No. 2, 2004, pp. 137-158.

[4] Biles, J.A., Anderson, P.G., & Loggi, L.W. (1996) *Neural network fitness functions for a musical GA*. In Proceedings of the International ICSC Symposium on Intelligent Industrial Automation (IIA'96) and Soft Computing (SOCO'96) (pp. B39-B44). Reading, UK: ICSC Academic Press.

[5] Khalifa, Y.M.A., Shi, H., Abreu, G., Bonny, S., and Ziender, J.: "Autonomous Evolutionary Music Composer", presented at the EvoMUSART 2005, held in Lausanne, March 2005.

[6] Horowitz, D.: Generating Rhythems with Genetic Algorithms. Proc. of the 12th National Conference on Artificial Intelligence, AAAI Press, 1994.

[7] Marques, M., Oliveira, V., Vieira, S. and Rosa A.C.: Music Composition Using Genetic Evolutionary Algorithms, Proc. of the Congress of Evolutionary Computation, Vol. 1, 2000.

[8] Pazos, A., and Del Riego, A.: Genetic Music Compositor. Proc. of the Congress of Evolutionary Computation, Vol. 2, 1999.

[9] Goldberg, D.E., Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, Reading , USA, 1989.