Tonality Driven Piano Compositions with Grammatical Evolution

Róisín Loughran
NCRA
UCD CASL
Belfield
Dublin 4, Ireland
Telephone: (+353) 1–7162692
Email: roisin.loughran@ucd.ie

James McDermott
NCRA
UCD CASL
Belfield
Dublin 4, Ireland
Telephone: (+353) 1–7162643
Email: jmmcd@jmmcd.net

Michael O'Neill
NCRA
UCD CASL
Belfield
Dublin 4, Ireland
Telephone: (+353) 1–7162654
Email: m.oneill@ucd.ie

***Abstract*—We present a novel method of creating piano melodies with Grammatical Evolution (GE). The system employs a context free grammar in combination with a tonality-driven fitness function to create a population of piano melodies. The grammar is designed to create a variety of styles of musical events within each melody such as runs, arpeggios, turns and chords without any a priori musical information in regards to key or time signature. The fitness of the individuals is calculated as a measure of their tonality defined by a statistical distribution of the pitches in each piece. A number of short compositions are presented demonstrating that our system is capable of creating music that is interesting and unpredictable.**

I. Introduction

This study applies an Evolutionary Computation (EC) method, specifically Grammatical Evolution (GE) to the task of composing a short novel piece of piano music. The application of EC and machine learning methods to subjective problems in creative domains such as art, music and humour have become more prevalent in recent years due to the complexity of measurement in these domains and the continuous increase in the capabilities of such computational methods. Computer programs can replace and outperform the human mind in computational tasks, but until a program can demonstrate true creativity, there is an argument that it cannot be deemed intelligent. By investigating these ill defined problem domains further, we hope to gain some possible insights into the fields of Artificial Intelligence or Computational Creativity.

Generative music or algorithmic composition is music that is created by an algorithm, set of rules or any given machine learning method. Such methods have been developed for creating music for over half a century [1]–[3]. Although we may think of generative music as being electronic or electro-acoustic, the term refers to the way in which music is produced or created and has no bearing on content, style or genre. Music can be generated by learning from previous music (data-driven) or by pure generation from a random seed, through a series of rules and grammars to a novel result. This study focusses on the second method.

Over the last few decades, EC methods have been applied in a number of ways to computationally creative systems such as music generation. The very nature of EC, creating a population of solutions before combining and mutating the more successful of these is generally not deterministic; a solution is rarely determined outright but approached from a number of locations. This is reminiscent of the music creation process — composition is not a linear, deterministic process, but a wandering combination of thoughts and decisions that, once started, would be unlikely to end up in the same position twice. Grammar-based generative methods, such as GE used in this paper, can be particularly suitable to generating music as it is a genome that is being manipulated rather than the piece of music itself. This allows the method to generate an output with a level of complexity far greater than with which it started. This added complexity generation can be be very helpful in creating interesting and diverse pieces of music. In the experiments proposed in this paper, the grammar defines the search domain — the allowed notes and musical events in each composition. Successful melodies are then chosen by traversing this search space according to the defined fitness function.

The following section discusses previous music composition studies that have incorporated EC techniques. Section III introduces the methods used in this experiment, in particular GE and measures of tonality. Section IV discuses the representation used within the system including the grammar and fitness function developed for our experiments. The experimental setup and a number of the compositions created are presented and discussed in Section V. Finally, some conclusions and future work are proposed in Section VI.

II. Previous Studies

There have been a number of previous studies that incorporate evolutionary methods with musical compositions. [4] offer an overview of earlier studies, determining that Genetic Programming methods preform better than those that use Genetic Algorithms. They state that the process of composition can be split into three stages: search domain, input representation and fitness measurement. The search domain is defined by a set of a priori constraints the user decides at the beginning of the composition — the notes allowed, the instrumentation used etc. Input representation refers to the way in which the music is represented for use by the system and fitness measurement is the manner in which the evolved piece is judged. While they argue these points for algorithmic and

978-1-4799-7492-4/15/$31.00

in particular evolutionary composers, it is clear that human composers also make similar judgements in regards to such criteria for a composition.

A number of studies to compose specific styles or aspects of music have incorporated evolutionary techniques. GenJam [5] used a GA to evolve jazz solos, building solos from pre-generated MIDI sequences that were judged by a user to determine the fitness measure. The system has been modified and developed into a real-time, MIDI-based, interactive improvisation system performance system that regularly performs in mainstream venues [6]. GeNotator is a composition tool presented in [7] that used a modified GA to manipulate a musical composition using a hierarchical grammar. Melody and rhythm were both evolved and evaluated separately using MLPs in [8]. These evolved melodies were then mixed to produce verses and whole songs. Dahlstedt developed a system that implements recursively described binary trees as genetic representation for the evolution of musical scores. The recursive mechanism of this representation allowed the generation of expressive performances and gestures along with musical notation [9]. The performance of EC algorithms on melody-matching tasks is examined in [10]. Although they do not create any new music, they examine a number of representations using GP including the use of Automatically Defined Functions (ADFs). They propose that ADFs may be useful in finding and reusing patterns within a melody. The results show that the inclusion of ADFs helped the system to find better approximations of the target melodies, indicating that the may be useful in representing melodies.

A. Interactive Evolutionary Computation

A large number of studies in involving creative compositions and EC use Interactive EC (IEC). These methods involve a human judgment either wholly or as part of the fitness evaluation. This type of fitness evaluation is very well suited to design and creative tasks as subjective judgements are extremely difficult to quantify. Interactive GE in particular has been used in a number of design tasks including pylon, truss and 3D design [11]–[13]. IGE and GE in general are well suited to such design problems as domain knowledge of the specified problem can be easily incorporated through the underlying grammatical representation. Such employment of domain knowledge is particularly useful in musical tasks. GE was used with an interactive fitness function for musical composition using the Wii remote for a generative, virtual system entitled Jive, [14]. This system interactively modifies a combination of piece-wise linear sequences to create melodic pieces of musical interest. [15] used GE for composing short melodies using four different experimental set ups of varying fitness functions and grammars. They used a combination of automatic fitness generation and interactive human judgment on a number of methods, determining that users preferred melodies created with a structured grammar.

The biggest drawback with interactive methods is that they create a bottleneck, particularly in musical tasks. For tasks such as visual analysis, whereby the user can observe a number of creations concurrently, the fitness can be evaluated very quickly. With musical judgments however, users need to listen to and concentrate on aural excerpts successively, rendering these methods expensive. Instead of using a human observer, the compositional method presented here autonomously creates musical excerpts from the ground up. Although interactive fitness has been shown to perform well and be informative, we focus on using an autonomous fitness function without the need for a human fitness evaluation. The measure of fitness may be difficult to quantify, but it is hoped that by developing judgement fitness functions in areas such as composition, we may learn more about how such judgements are made and about computational creativity itself. Using a few simple rules from a defined grammar, we create melodic pieces that are not trained from previous pieces, are not judged by a human listener and are not given any a priori musical information such as key signature or time signature. As such, the composed pieces vary significantly and have an ad-hoc, organic nature to them, as discussed later in this paper.

III. Method

The proposed method uses a python implementation of GE entitled PonyGE to evolve or compose musical pieces. PonyGE evolves a phenotype representing a MIDI sequence of notes which is interpreted and played through GarageBand.

A. Grammatical Evolution

Grammatical Evolution (GE), [16], [17] is a grammar based evolutionary computation method. As with other EC algorithms, GE is based on the creation of a population of solutions or individuals that are evolved over a number of generations using operators such as selection, mutation and crossover. GE employs a user-defined context-free grammar to map the evolved genotype of an individual to its phenotype. The fitness of the individual can then be evaluated from a measure of the phenotype's success at solving the given problem. Typically the grammars used are in Backus-Naur Form (BNF).

1) Grammar: The creative capabilities of GE come from the choices offered within the mapping of the grammar. Typically, the genome is represented by a combination of 8 bit integers known as *codons*. These codons select the particular rule for a given expression according to the mod value from the number of choices for that rule.

$$\text{Rule} = (\text{Codon Integer Value}) mod (\text{\# of choices}) \quad (1)$$

Using this we can introduce biases by including multiple instances for preferred choices. For example, the operand depicted in Equation 2 offers three choices, two of which are choice1. Thus there is a 2:1 bias towards the selection of choice1 over choice2. We make use of such biases in our experiments to incorporate our knowledge of the musical domain.

$$< operand > ::= < choice1 > | < choice1 > | < choice2 > \quad (2)$$

2) Representation: We exploit the representational capabilities of GE resulting from the definition of a grammar that can explore the given search domain. While many studies involving GE generate program code, the grammar can map the phenotype to contain any character or integer string the user requires. This phenotype can then be interpreted by the user in a predetermined manner. In these experiments, we use integer strings to represent sequences of MIDI notes. We create a number of grammars that can map to specific integer values which are in turn interpreted into note attributes such as pitch, duration and style.

B. Tonality

Most tonal music contains some sense of tonal hierarchy. Certain tones are given preference and are repeated and emphasised throughout the piece giving the music a sense of stability. In western tonal music this is generally based on the major or minor key signature of the piece but the details of tonal hierarchy differ across styles and cultures. This variation suggests that such hierarchies are based less on the acoustic relationship between the harmonic structures of complex tones but that an expected tonality is more reliant on perception and cognition [18]. Krumhansl defines Tonal Induction in [19] as the process by which a listener identifies the key of a given piece of music. Although originally considered for Western tonal music there is evidence shown, for example in [20], that when given a repeated tone, Westerners can similarly induce pitch keys from unfamiliar music such as Indian music. This implies that there may be no 'correct' tonality, but rather a tonal hierarchy could be developed or induced by a new system with no a priori knowledge of tonal relationships and hence no pitch expectations.

Similarly, [21] discusses the Distributional view of key identification — that the perception of key depends on the distribution of pitch-classes within a piece. Such studies imply that all that is required to induce or infer a perceived pitch key is to enforce a preference or repetition of a given set of pitches. The current study does not use major or minor tonal keys as defined by Western tonality, but rather we try to enforce a ratio between the used pitches so that a certain number of pitches appear much more frequently than others.

The driving force behind any evolutionary method is the fitness function. For subjective tasks, such as musical composition, the definition of what makes an individual more fit than another is ill-defined. This is the strongest argument for employing IEC in such tasks. For this study however, we wanted to use a more autonomous approach. In the proposed experiments, we use a measure of emergent tonality as the fitness evaluation for evolved individuals. Rather than giving a key signature, we reward compositions that induce their own sense of tonality. Compositions that emphasise a certain number of pitches within the scale over others will be considered more tonal and hence judged more fit than others that show an even distribution of pitches. Thus we drive a statistical preference to the use of a certain number of pitches but make no assertions as to the relationship between prevalent pitches. We propose that this will in turn induce a key or tonality to a given piece, albeit not one we would be used to.

IV. Representation

A. Grammar

The BNF Grammar controls the way in which the evolved piece of music progresses over time. Our grammar maps the genome (a combination of 8 bit codons) to the phoneme which represents melodies as a combination of events entitled notes, chords, runs, turns and arpeggios as defined by the grammar. The initialisation of the piece is given by the line

```
<piece> ::= <event>|<piece><event>|
            <event><piece><event>|
            <event><piece><event><event>
```

This creates a piece of music <piece> that is comprised of a sequence of musical events <event>. A BNF grammar such as this offers a number of choices for each mapping separated by a |. Thus the first two options alone would suffice in expanding out a melody. The inclusion of the last two options is to allow the melody to expand in length in early generations. PonyGE uses random initialisation, therefore if only two initial options are given it can take a number of generations for the genome to grow to a useful size. The expansion of the remainder of the grammar is based on a series of musical events, here denoted by <event>:

```
<event> ::= 111,<style>,<oct>,<pitch>,<dur>,
```

The constant 111 indicates the beginning of a new <event>. This allows the resultant phenotype to be correctly interpreted into individual note events. Every <event> is attributed a given value for style, octave, pitch and duration. Each of these attributes expand uniquely within the grammar in relation to what they represent. These expansions are detailed below.

Pitch is simply a value between 0 and 11 chosen with equal probability to represent which of the 12 pitches in the chromatic scale this note event is based on. Octave refers to the octave number the current event starts in. Although the pitch of a piano can range across 8 octaves, we would like to keep the majority of the notes played within the mid range of the piano for the moment, to encourage simple melodies. Hence <oct> is expanded to:

```
<oct> ::= 3 | 4 | 4 | 4 | 4 | 5 | 5 | 5 | 6 | 6
```

This means that only notes in octave 3-6 will be allowed in any piece, and that notes in the 4th and 5th octave will be given preference. This encourages the melody to remain around middle C (Octave 5, pitch 0) but it allows notes to be played in higher and lower octaves. The use of biases such as this is one of the practical benefits of using a grammar for melodic composition and is exploited many times in our system.

Rather than specifying a rhythm or timing for the melody, we allow each note to have a specific duration. The attribute for duration is given by:

```
<dur> ::= 1 | 1 | 2 | 2 | 2 | 2 | 4 | 4 | 4
  | 8 | 8 | 16 | 16 | 32
```

This line of the grammar defines the duration of the note event from a demisemiquaver (value 1) to a semibreve (value 32). As with the octave attribute, a bias is introduced to encourage shorter notes within the melodies with notes shorter than a quaver (value 4) given more instances and hence higher preference over longer minim and semibreve notes. The inclusion of longer notes is very important in music, but melodies that consist of mostly long notes can be dull.

The <style> attribute defines the current type of note event. The type of events available to the grammar are note, chord, turn and arpeggio:

```
<style> ::= 100 | 100 | 100 | 100 | 100 | 100 |
   100 | 100 |  50,<chord> |  50,<chord> |
```

```
50,<chord> |  50,<chord> | 70,<turn>,100 |
80,<arp>,100
```

The style of each event is determined by the constant value preceding it. As can be seen from the above code, the value `100` signifies a plain note, `50` represents a chord, `70` a turn and `80` an arpeggio. Notes and chords are single events taking less time to play but can be more pivotal to the overall piece than turns and arpeggios. Hence there are more instances of chords and particularly notes in this grammar. A `100` (note) requires no further information than the octave, pitch and duration already assigned to it and hence requires no further grammar. A chord is defined by the pitch value already given to it and the inclusion of either one, two or three notes played in conjunction with it:

```
<chord> ::= <int>, 0, 0 | <int>,<int>, 0 |
      <octave>,0 ,0 | <int>,0, 0 | <int>,0, 0 |
      <int>, 0, 0 | <int>, <int> , <int>
<octave> ::= 12
<int> ::= 3 | 4 | 5 | 5 | 5 | 7 | 7 | 7
```

To discourage overly-heavy chords, those with only one upper note `<int>,0,0` were encouraged above all others. The `<int>` values indicate the number of semitones above the tone that is played with it. Hence the above grammar allows interval of a minor third, major third, perfect fourth, perfect fifth and the octave, with a preference for perfect intervals.

A turn is defined as a short run of notes in one direction (up or down) followed by another short run in another direction:

```
<turn> ::= <dir>,<lenT>,<dir>,<lenT>,<stepDur>
<dir> ::= 45 | 55
<lenT> ::= <step> | <step>,<step>
       | <step>,<step>,<step>
       | <step>,<step>,<step>
       | <step>,<step>,<step>,<step>
<step> ::= 1 | 1 | 1 | 1 | 1 | 2 | 2 | 2
       | 2 | 2 | 2 | 2 | 2 | 2 | 3
<stepDur> ::= 1 | 2 | 2 | 2 | 2 | 2 | 2 | 4
       | 4 | 4 | 4 | 4 | 4 | 4
```

The direction up or down is chosen at the beginning and again halfway through the turn. As the second choice of direction is independent from the first, this grammar will produce a run (both directions the same) 50% of the time, resulting in no need for a separate grammar line for runs. The length of each section of the turn is one, two, three or four notes with a bias towards three. Each step size within the turn is either one or two semitones, with the occasional allowance of a step of three semitones. The duration of the step is limited to either semiquavers or quavers. The grammar for the arpeggios is very similar to this for a turn, except that the allowed step size is `<int>` as defined earlier in `<chord>` and slightly longer durations are allowed.

In GE, the genetic operators manipulate the genotype, while fitness evaluation is taken from the corresponding phenotype. Thus a small mutation can result in a large change in the phenotype and hence the melody. This is one of the main advantages in using GE; the grammar mapping, such as the one described above, adds a level of complexity that can encourage change and variety within the music.

B. Fitness Function

The above grammar maps a genotype to it's corresponding phenotype. The phenotype is a string containing a list of numbers that are in turn interpreted into a musical context. The fitness of an individual is measured from the distribution of the pitches among the melody in combination with the length of the phenotype. The function, described below, is a minimising fitness function.

The initial fitness of an individual is set in relation to the size of the current phenotype in comparison to a given constant. For these experiments this constant was set to 200. In this way, GE can quickly create a phenotype of a specified length. This length is related to the length of the produced piece of music via the grammar. It does not, however, directly control the length of the piece of music produced, or even the number of notes included, due to the use of turns and arpeggios within the grammar. From the grammar, a single note event will contain five values, a chord will contain eight values and a turn or arpeggio can contain up to seventeen values depending on the length of the turn. The initial fitness is calculated as:

$$fitness_{initial} = (Len - 200)^2 + 1 \tag{3}$$

where Len is the length, in integer values, of the current phenotype. The addition of the constant 1 is to prevent a fitness of zero as this initial fitness is now adjusted by multiplication according to the statistical relationship between the pitches.

The fitness of each individual is measured in relation to the perceived tonality of the piece. As described in Section III tonality is largely perceived in response to repetition of pitches. Thus we measure the fitness of a given individual based on the ratio of the pitches within the current phenotype. The number of instances of each pitch in the individual is summed and the instances of pitches are arranged in decreasing order. Only `<event>` pitches as defined by the grammar are considered for this measurement — pitches introduced due to the turns or chords in the grammar are disregarded.

For a true emergent tonality we require one pitch to be the most frequently played within the melody, and we require an unequal distribution of the remaining pitches. In our fitness we define the *primary* as the pitch value with most instances and the *secondary* as that with the second highest number of instances. Thus for a good (low) fitness we want the number of primary pitches to be significantly higher than the number of secondary pitches.

If,

$$\frac{\#\text{ instances of primary}}{\#\text{ instances of secondary}} < 1.3 \tag{4}$$

the fitness is multiplied by a factor of 1.3.

This enforces the primary tone to have significantly more instances than the secondary, encouraging one strong pitch within each melody. We further define the number of instances of the seven most frequently played notes as Top7 and the number of instances of the top nine notes as Top9.

If,

$$\frac{\text{Top7}}{\text{Total number of played notes}} < 0.75 \quad (5)$$

or if

$$\frac{\text{Top9}}{\text{Total number of played notes}} < 0.95 \quad (6)$$

the fitness is again multiplied by 1.3.

This encourages most of the played notes to be within the top seven or top nine notes, but still allows for accidentals to be played. In uniformed pitch-distributed music these ratios would work out to be 58% and 75% respectively whereas in Western tonal music all notes, bar accidentals, would be contained in the top seven pitches. Thus, with our limits of 0.75 and 0.95 we are encouraging more tonality than 12 tone serialism but not enforcing the limits of Western tonality. This fitness function is completely reliant on the statistical measure of the frequency of notes played within the piece. It does not take into account the order in which the notes are played. Also, the fitness is based on the relationship of the instances of the notes but has absolutely no bearing of the relationship between the pitches themselves. In this regard it is highly unlikely to produce a similar key to one from the Western tonal system.

V. Experiment and Results

The goal of our experiment is to evolve musical piano melodies without the need for human interaction. The experiments were run with a population of 100 for 50 generations. For these experiments all other parameters were left to the default settings in PonyGE: the mutation coefficient was set to 0.01, crossover was set to 0.7 and there was an elite size of 1. In comparison to many EC studies, these experimental runs are very short, taking less than 30 seconds to complete. A number of the evolved compositions are available at the following address: http://ncra.ucd.ie/Site/loughranr/music.html. The production and validity of these melodies are discussed below.

A. Short Melodies

Our first melodies were created from single individuals evolved using the grammar and fitness function described above. For these runs the target length of the phenotype, Len in Equation 3, was set to 200. Individual1.mp3 is a typical example of a melody created by the system. This melody demonstrates all of the note events that the grammar is capable of creating — notes, chords, turns and arpeggios. The primary key emergent from this melody is E, with four instances of this note in comparison to three instances of the secondaries $F\#$, G and $A\#$.

From listening to this composition it is clear that it is musical, but somewhat meandering. This is true for many of the evolved compositions, notably Individual3.mp3. While there does still appear to be some tonality to this composition, pitch $A\#$ is sternly primary with seven instances compared to only four for the next three secondary pitches, there is a wandering quality to the piece. This is to be expected as the grammar does not encourage any repetition or variation on a theme but creates independent events sequentially. Such compositional aspects can be highly desirable at times, if the piece is to create an air of suspense for example. There is the danger, however, that such pieces can end up sounding unfinished and meandering, with no real purpose or direction.

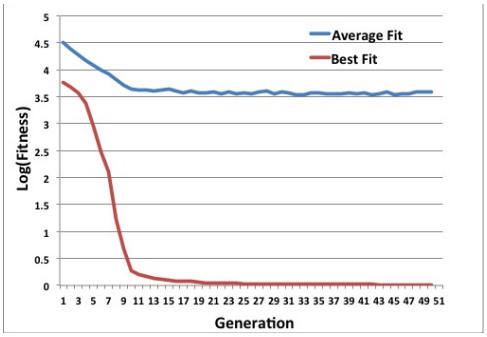


Fig. 1: Average and Best Fitnesses over 30 runs. Fitness values reported as $log_{10}(Fitness)$.

In some cases, the melody can be overshadowed by too many fast runs and arpeggios as in Individual2.mp3. This can make the melody overly complex and very fast moving. Again, sometimes in a composition this may be desirable, but if not we can try to curb such behaviour by increasing the bias even further in the grammar towards notes and chords rather than turns and arpeggios.

1) Fitness Evaluation: In any evolutionary run, it is useful to track how the best fitness evolves in relation to the average fitness of the population. To examine this, we ran our system 30 times and took the mean of the best and average fitnesses over the 50 generations. A plot of the log of the results is given in Figure 1. It is clearly evident from these runs that although the best fitness converges quite quickly to the target (within approximately 12 generations), at the end of the run the average fitness is still very high. This implies that after 50 generations the population is still very diverse. Generally in EC tasks we would like the population to converge on the correct solution. This diversity indicates that the fitness measure currently used is not allowing the population to converge and that the individual melodies in the final generation are very different from one another.

To examine this diversity within the final generation we recorded the individual fitness values, phenotype lengths and Top7 and Top9 values for each of the 100 individuals in the final generation of a run. A plot of each of these values is shown in Figure 2. It can be seen from the fitness plot that while the majority of individuals achieve low fitness, 11 individuals have fitness values over 10,000 with one individual approaching 100,000. Such abnormally high fitnesses will inevitably push up the average fitness throughout the run. It can be seen from the phenotype length plot that each of these high fitnesses correspond to a particularly long or short length. As the initial fitness is calculated from the square of the distance between the phenotype length to a given constant, this correlation is not unexpected. The graphs of Top7 and Top9 for each individual show that while the majority can achieve the target of 0.75 for Top7, only a small number manage to have their Top9 pitches to within 95% of all notes played. This indicates there are still a high number of melodies containing all (or almost all) 12 pitches. From an examination

of the relationship between the primary and secondary pitches we found that the ratio between them for most individuals did not exceed the target of 1.3. This analysis of the fitness components indicates that to enforce a stronger tonality we could introduce higher penalties for individuals that perform weakly on the primary to secondary ratio and the Top9 value.

To further examine a good versus bad individual, we considered the pitch distribution between two specific individuals. Figure 3 displays the pitch distribution of the best and worst individual in the final generation of an evolutionary run. These bar charts indicate the differences in distribution of pitches in these individuals. Figure 2(a) shows the pitches in Individual3.mp3 which achieves an perfect fitness of 1. From this graph there is a clear primary pitch that is more present than the others, with three secondary pitches standing out and only the top nine pitches featuring in the melody. The length of the phenotype is 200 giving a perfect initial fitness of 1. Further to that there are a total of 31 pitches played with 25 within the top 7 and 30 within the top 9 giving distribution percentages of 81% and 97% respectively. In contrast, Figure 2(b) displays the pitches in WeakIndividual.mp3 whose fitness is 34,330. This particular individual has a high (bad) fitness in every measure. The length of the phenotype is 325, giving it an initial fitness of over 15,600. From Figure 2(b) it can be seen that there are two equally strong primary notes and that every pitch is represented in the melody at least once resulting in the percentage of notes in the top 7 and top 9 at only 67% and 80% respectively. Hence the initial bad fitness is increased at each measurement. This WeakIndividual.mp3 can be listened to at http://ncra.ucd.ie/Site/loughranr/music.html. What may be surprising is that this melody does not immediately sound any 'worse' than Individual1. It is heavy on the runs, which would naturally increase the length of the phenotype but the presence of each chromatic tone and the lack of a primary key does not appear obvious on the first listen. As discussed in Section III-B however, it is the repetition and emphasis of such patterns that can cause perceptual keys to emerge. This idea led us to combine a number of fit individuals together as described in the next section.

From an inspection of a number of weaker individuals in different runs, it was clear that many had high (bad) fitness due to their lengths either being significantly too short or too long. This implies that the initial weighting of squared distance from the target to actual length is too strong for the entire population to overcome. Even so, in every run the majority of individuals achieved good fitness with a number of highly fit or perfect individuals emerging within 11 generations, indicating that it can be a good driving force. It is worth noting again that enforcing a given phenotype length does not directly force the melody to be a certain duration or even a certain number of notes in length. The mappings in the grammar can lead to events of different length within the phenotype, hence the length of the piece of music is not strictly enforced by this fitness function but can be within a given range. In future work we may consider having a range of possible lengths so an individual is only heavily penalised if it is significantly outside this range. Furthermore, the fitness function may easily be altered if we wish to drive the evolution according to another property such as rhythm, melody progression or a more subjective fitness measure, should it be proposed.

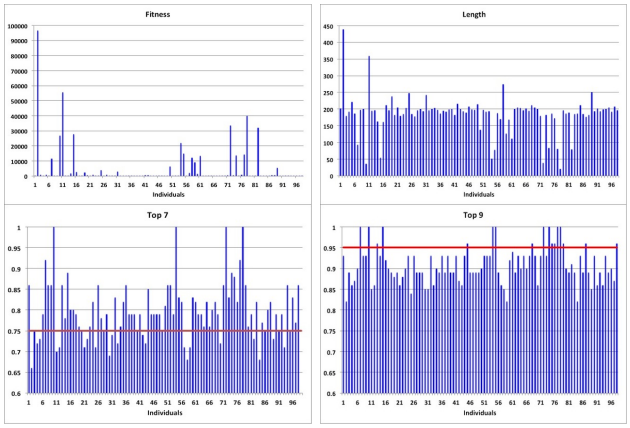
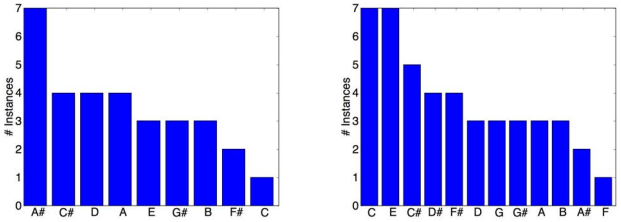


Fig. 2: Fitness, Length, Top7 and Top9 for each of the individuals in the final Generation



(a) (b)

Fig. 3: Pitch Distribution for a (a) highly fit individual displaying one prominent pitch and a tail that tapers off after nine pitches, (b) weak individual with two equally prominent pitches and all 12 pitches present

B. Longer Melodies

The compositions created by the above method are very short. The fitness function ensures a short phenotype length which in turn creates short pieces. There is also little form or regularity within the piece. Temporally, the progression of melody does not depend on any previous notes, and no particular themes or motifs emerge. A particularly useful feature of EC methods in this regard is that we have evolved a population of solutions — a number of highly fit individuals that are similar but not identical. As we have seen above, the population of solutions even after 50 generations remains diverse, indicating that the final population contains varied melodies. Thus instead of merely taking the best individual, we can take a selection of the best individuals and splice them together. Even in cases where the fitness has reached a minimum (local or global) and a number of top individuals have equal fitness, the individuals themselves are rarely identical. This idea of splicing individuals together to create a composition was used by Rodney Waschka II in his program GenDash [22]. This program combined all individuals from successive generations in creating entire compositions or aspects of compositions. Although we only combine a small number of individuals from the final population, we found that combining individuals in this manner led to a number of interesting compositions.

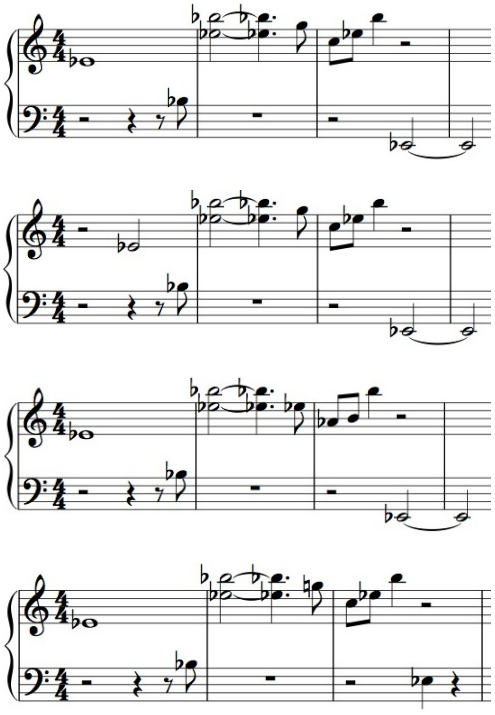


Fig. 4: Motif and three variations emergent from Melody 1

Melody1.mp3 is a good example of combining numerous similar but non-identical individuals that enables a motif to emerge from a melody. This composition was created by combining the top four individuals from the final generation of an evolutionary run. From listening to this composition, the semibreve $E\flat$ is immediately obvious as a striking yet simple repetition within the piece. In this example, this particular note indicates the beginning of the individual, but the remainder of each individual differs causing variations within the theme. Figure 4 displays scores of the first part of each individual that share a variation on a motif. The original theme, in the top score, is varied slightly in regards to rhythm and pitch but is distinctly recognisable each time it is played. The remainder of each individual varies considerably by comparison but the return to this motif anchors the listener and gives the composition a sense of form. This recurrence of similar content is due to selection and crossover among highly fit individuals; the first part of the melody has survived to be recreated in a number of the top individuals. Other short motifs emerge that are more subtle yet clearly audible within the piece. Similar emergent themes and motifs can be heard clearly in Melody2.mp3 and Melody3.mp3.

C. Discussion

It is immediately evident from working with this system that there is a large variation in the compositions being produced. We would argue that this variation is more of a testament to the success of the system rather than a flaw within it. If the system reproduced the same or very similar melodies every time it was run, there would be no compositional benefits to using it at all. While the variation between runs is high, the motifs that emerge from the best individuals in the last generation show that the system is capable of focussing on and refining one given idea. If we are use such a system as a computational aid, something to give the working composer new ideas, then diversity and variation among the produced musical excerpts, while having some confidence in the merit of the music produced is the ideal goal of such a system.

In saying that, we acknowledge there are limitations in the compositions currently produced. As the compositions stand, they lack in their overall form. Although the inclusion of four individuals can introduce perceived repetitions and motifs within the piece, the majority of the melodies produced still show little in terms of beginning, progression and ending. This cannot be immediately addressed by the system as it works at the moment. The melodic progression is independent of previous notes and so cannot influence the development of the music. Likewise, the ordering of the individuals in the longer melodies is not related to their progression or their note content, but merely based on their fitness. As long as the fitness is entirely dependent on the statistical tonality of the evolved melody it will not be able to influence the progression of the composition. To fully address this issue, we would need to create a system that takes into account the form of the composition. Until the system can recognise and respond to the progression of the melody it is unlikely to be able to create fully formed, concise compositions. At the moment, it is evident that the system cannot create a full composition, but we do see it as an interesting compositional tool, one which can aid a composer in initialising ideas.

As discussed, this system responds to a statistical fitness function of distribution of pitches, it does not guarantee a pleasant melody. As an extreme, let us consider a melody that consists of 40 quaver notes, 30 of which are played at pitch C, octave 2 followed by the remaining 10 played at C#, octave 2. The initial fitness would be set to 1 as there are 5 values per single `<note>` in the phenotype, thus 40 notes would give a perfect initial fitness of 1. Furthermore, C emerges prominently as the primary key with $C\#$ as the notable secondary and the ratio in equations 5 and 6 would both be 1 leading to an optimal fitness value of 1. Clearly this is not a good melody [1]. Although theoretically our fitness function would give this contrived melody perfect fitness, in our system, the grammar would be highly unlikely to ever generate such a composition. This fitness function is not intended as a universal measure of the quality or 'goodness' of the melody, but it is a statistical measure that can quantify a given attribute (in this case tonality) as a method of guiding the algorithm through the search space created by the grammar. This statistical measure of pitches is similar to an approximation of the power law or Zipfs distribution. Zipfs distribution has been used on a number of musical attributes in classifying music in terms of aesthetic and 'pleasantness' [23]. In future development of this project, we plan to extend the fitness function to emanate this power law distribution to better evaluate the fitness of an individual.

Ideally, we wish to judge the merits of a compositional system on the aesthetic quality of it's output. Any aesthetic judgement is highly subjective and therefore extremely difficult to quantify and measure. We are tentatively pleased with the quality of a number of the compositions produced by our

[1] although we must point out that with a slight rearrangement to interpolate the pitches this does emanate the theme from 'Jaws'

system as we find them to be interesting, vibrant and at times fun. At the moment, the system will not automatically create a masterpiece, but it does create snippets of original music which we look forward to developing in the areas of form and evaluation. As a creative system, it stops shy of addressing open problems in evolutionary art and music such as those proposed in [24]. Nevertheless, we have created a system that can create novel sections of music using a new representation and grammar in GE with a tonality-driven fitness function. We hope to push this system further into the domain of evolutionary music and computational creativity by developing a way to measure the true aesthetic value or merit of the evolved melodies.

VI. Conclusion and Future Work

A number of original piano melodies created in GE with a user-defined grammar and tonality-driven fitness function have been presented. We described our domain specific grammar that maps from genotype to phenotype to create complex melodies containing runs, turns and arpeggios. These were evolved with a fitness function based on the number of pitches present in the melody to create a variety of compositions. These individual compositions were in turn combined together to create longer pieces of music that contain discernible patterns and motifs.

Piano music is conventionally written on two staves, one for the treble and one for the bass. Although a number of the presented pieces would be played by two hands on a piano, it is one melody line that is being evolved. We are currently looking at methods to evolve a bassline that would accompany a given melody in both pitch and rhythm. We are also currently looking at ways to expand the grammar so that it may consider the form of a full composition. Thus our system would be able to produce a piano piece more akin to a typical piece played by a human performer.

The fitness used is a measure of tonal statistical validity, rather than a true measure of creativity or even musical 'goodness'. As such, our system may not always create a particularly noteworthy or pleasant sounding piece; each composition may not be to everyone's tastes, but more often than not they are of at least some musical interest. As future work, we are very interested in developing a fitness function that is able to make a more subjective judgement of music — not one that would judge a certain style, but one that could correctly identify or discern musical merit. Creating an aesthetic fitness function that can reliably judge the merits of any creative object is still an extremely difficult, ill-defined problem. We hope that by developing a more automated system, we may be able to investigate this area more thoroughly and in turn gain insights into the usage and capabilities of computational creativity.

Acknowledgment

The authors would like to thank fellow members of the Natural Computing Research and Applications group (NCRA) for their insightful discussions and recommendations in regards to this paper. This work is part of the App'Ed (Applications of Evolutionary Design) project funded by Science Foundation Ireland under grant 13/IA/1850.

References

[1] L. A. Hiller Jr and L. M. Isaacson, "Musical composition with a high speed digital computer," in *Audio Engineering Society Convention 9*. Audio Engineering Society, 1957.

[2] I. Xenakis, "Formalized music: thought and mathematics in composition. harmonologia series," 1992.

[3] B. Eno, "Generative music." [Online]. Available: http://www.inmotionmagazine.com/eno1.html

[4] A. R. Burton and T. Vladimirova, "Generation of musical sequences with genetic techniques," *Computer Music Journal*, vol. 23, no. 4, pp. 59–73, 1999.

[5] J. Biles, "Genjam: A genetic algorithm for generating jazz solos," in *Proceedings of the International Computer Music Conference*. INTERNATIONAL COMPUTER MUSIC ASSOCIATION, 1994, pp. 131–131.

[6] J. A. Biles, "Straight-ahead jazz with GenJam: A quick demonstration," in *MUME 2013 Workshop*, 2013.

[7] K. Thywissen, "Genotator: an environment for exploring the application of evolutionary techniques in computer-assisted composition," *Organised Sound*, vol. 4, no. 02, pp. 127–133, 1999.

[8] H. Göksu, P. Pigg, and V. Dixit, "Music composition using genetic algorithms (GA) and multilayer perceptrons (MLP)," in *Advances in Natural Computation*. Springer, 2005, pp. 1242–1250.

[9] P. Dahlstedt, "Autonomous evolution of complete piano pieces and performances," in *Proceedings of Music AL Workshop*. Citeseer, 2007.

[10] A. Pirnia and J. McCormack, *Compressed multidimensional trees for evolutionary music representation*. Ann Arbor, MI: MPublishing, University of Michigan Library, 2012.

[11] M. O'Neill, J. McDermott, J. M. Swafford, J. Byrne, E. Hemberg, A. Brabazon, E. Shotton, C. McNally, and M. Hemberg, "Evolutionary design using grammatical evolution and shape grammars: Designing a shelter," *International Journal of Design Engineering*, vol. 3, no. 1, pp. 4–24, 2010.

[12] M. Fenton, C. McNally, J. Byrne, E. Hemberg, J. McDermott, and M. O'Neill, "Automatic innovative truss design using grammatical evolution," *Automation in Construction*, vol. 39, pp. 59–69, 2014.

[13] J. Byrne, J. McDermott, E. Galván-López, and M. O'Neill, "Implementing an intuitive mutation operator for interactive evolutionary 3d design," in *Evolutionary Computation (CEC), 2010 IEEE Congress on*. IEEE, 2010, pp. 1–7.

[14] J. Shao, J. McDermott, M. O'Neill, and A. Brabazon, "Jive: A generative, interactive, virtual, evolutionary music system," in *Applications of Evolutionary Computation*. Springer, 2010, pp. 341–350.

[15] J. Reddin, J. McDermott, and M. O'Neill, "Elevated pitch: Automated grammatical evolution of short compositions," in *Applications of Evolutionary Computing*. Springer, 2009, pp. 579–584.

[16] M. O'Neil and C. Ryan, *Grammatical evolution*. Springer, 2003.

[17] I. Dempsey, M. O'Neill, and A. Brabazon, *Foundations in grammatical evolution for dynamic environments*. Springer, 2009, vol. 194.

[18] C. L. Krumhansl and L. L. Cuddy, "A theory of tonal hierarchies in music," in *Music perception*. Springer, 2010, pp. 51–87.

[19] C. L. Krumhansl, "Tonality induction: A statistical approach applied cross-culturally," *Music Perception*, pp. 461–479, 2000.

[20] M. A. Castellano, J. J. Bharucha, and C. L. Krumhansl, "Tonal hierarchies in the music of north india." *Journal of Experimental Psychology: General*, vol. 113, no. 3, p. 394, 1984.

[21] D. Temperley and E. W. Marvin, "Pitch-class distribution and the identification of key," *Music Perception*, vol. 25, pp. 193–212, 2008.

[22] R. WASCHKA II, "Composing with genetic algorithms: Gendash," in *Evolutionary Computer Music*. Springer, 2007, pp. 117–136.

[23] B. Manaris, J. Romero, P. Machado, D. Krehbiel, T. Hirzel, W. Pharr, and R. B. Davis, "Zipf's law, music classification, and aesthetics," *Computer Music Journal*, vol. 29, no. 1, pp. 55–69, 2005.

[24] J. McCormack, "Open problems in evolutionary music and art," in *Applications of Evolutionary Computing*. Springer, 2005, pp. 428–436.