Grammatical Evolution with Zipf's Law Based Fitness for Melodic Composition

Róisín Loughran
NCRA, UCD CASL,
Belfield, Dublin 4
roisin.loughran@ucd.ie

James McDermott
NCRA, UCD CASL,
Belfield, Dublin 4
jmmcd@jmmcd.net

Michael O'Neill
NCRA, UCD CASL,
Belfield, Dublin 4
m.oneill@ucd.ie

ABSTRACT

We present a novel method of composing piano pieces with Grammatical Evolution. A grammar is designed to define a search space for melodies consisting of notes, chords, turns and arpeggios. This space is searched using a fitness function based on the calculation of the Zipf's distribution of a number of pitch and duration attributes within the given melodies. In this way, we can create melodies without specifying a key or time signature. We can then create simple accompanying bass parts to repeat under the melody. This bass part is evolved using a grammar created from the evolved treble line with a fitness based on Zipf's distribution of the harmonic relationship between the treble and bass parts. From an analysis of the system we conclude that the designed grammar and the construction of the compositions from the final population of melodies is more influential on the musicality of the resultant compositions than the use of the Zipf's metrics.

1. INTRODUCTION

Music composition is a complex, aesthetic process. In recent years many composers, musicologists and computer scientists have looked to machine learning, autonomous methods of creating music either in conjunction with, or instead of the traditional human composer. We present one such study in which we employ an Evolutionary Computation (EC) method, namely Grammatical Evolution (GE) in the composition of piano pieces.

GE [1, 2] offers a versatile way of accessing and searching through a problem while taking advantage of problem domain knowledge. GE has been shown to be effective at a wide range of creative tasks including pylon and truss design, navigation in computer games and graphical logo design [3–6]. EC methods in general are not deterministic; a solution is rarely determined outright but rather approached from a number of locations. This makes them particularly suitable to aesthetic problems such as musical composition — composition is not a linear, deterministic process, but a combination of decisions that, once started, would be unlikely to end up in the same position twice.

This paper discusses the representations, grammars and fitness functions that we use to employ GE as an autonomous composer of piano pieces.

The following section details some previous experiments in using EC methods to compose music. Section 3 introduces Grammatical Evolution and gives a background the Zipf's Law power distribution used throughout this study. Section 4 details the workings of the experiment: the grammar used, the fitness measured and the manner in which we create an accompanying bass part. Section 5 presents and discusses and number of the melodies created by the system. Some conclusions and future work are proposed in Section 6.

2. PREVIOUS WORK

A number of previous studies have employed EC techniques for melodic composition. One of the most successful and well-known applications is GenJam [7] which uses a Genetic Algorithm (GA) to evolve jazz solos. This system has been modified and developed into a real-time, MIDI-based, interactive improvisation system that is regularly used in live performances in mainstream venues [8]. A modified GA was used in GeNotator [9] to manipulate a musical composition using a hierarchical grammar. Göksu et al. evolved and evaluated both melody and rhythm separately using MLPs [10]. 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 [11].

GE was first used for musical composition by de la Puente et al [12]. They tested the use of GE to generate melodies for a specific processor but did not present or discuss the melodies produced. More recently GE has been implemented for composing short melodies in [13]. From four experimental setups of varying fitness functions and grammars they determined that users preferred melodies created with a structured grammar. GE was again employed for musical composition using the Wii remote for a generative, virtual system entitled Jive [14]. This system interactively modifies a combination of sequences to create melodic pieces of musical interest.

Most of the above methods employ Interactive EC (IEC) methods, whereby a human observer is used within the fitness function. While a human observer is ideal for making subjective judgments on aesthetic processes such as art

and music, IEC is extremely costly. In the proposed experiments we avoid IEC and instead opt for an autonomous evaluation of the individuals based on Zipf's Laws.

Zipf's Law has been used in the investigation of pleasantness in music [15] and has been used previously as a fitness function in EC [16]. Zipf's Law relates to the frequency of occurrence of events and has been shown to turn up in many aspects of nature [17]. Formally, Zipf's Law states:

$$P(f) \sim 1/f^n \quad (1)$$

where P(f) is the probability of an event whose ranked frequency of occurrence is f and where n is close to 1. The number of occurrences are noted for each type of event. These occurrences are plotted against their statistical rank on a log-log scale. For an ideal Zipf's distribution we expect all points to fall on a straight line with a slope of -1. The R^2 value is a measure of how much the given points conform to this line, ranging from 0 to 1 with 1 denoting a straight (ideal) line. In order to calculate the Zipf's fitness for an attribute within a given individual (melody), we calculate the slope and R^2 of the rank-frequency distribution of this attribute and compare it to these ideal values.

The contribution of this study to the field of algorithmic composition lies in the exploitation of GE's capabilities to use grammars in representing and manipulating musical phrases. We use the population aspect of GE to combine multiple highly fit individuals together. We then use a two-run process where the first run evolved a treble melody, a new grammar is dynamically created in response to it, and then the second run uses this grammar to evolve an accompanying bass line. At the end of the study, we examine the resultant melodies in relation to each of the aspects used in creating them.

3. GRAMMATICAL EVOLUTION

GE is a grammar based algorithm based on Darwin's theory of evolution. As with other evolutionary algorithms, the benefit of GE as a search process results from it's operation on a population of solutions rather than a single solution. From an initial population of random genotypes, GE performs a series of operations such as selection, mutation and crossover over a number of generations to search for the optimal solution to a given problem. A grammar is used to map each genotype to a phenotype that can represent the problem under investigation. The success or 'fitness' of each individual can then be assessed as a measure of how well this phenotype solves the problem. Successful or highly fit individuals reproduce and survive to successive generations while weaker individuals can be weaned out. Such grammar-based generative methods 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 the original input. This added complexity generation is 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.

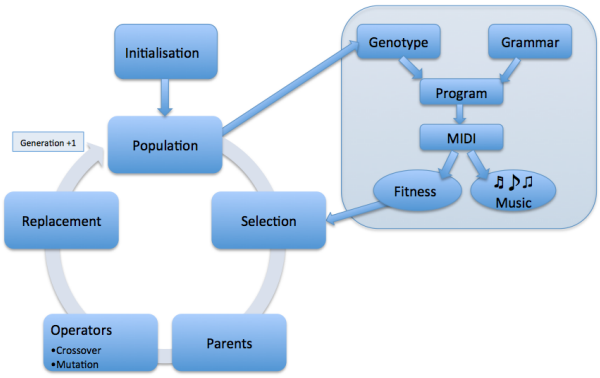

Figure 1: Overview of GE Process.

The creative capabilities of GE come from the choices offered within the mapping of the grammar. The grammars in GE are used to map the genotype to the phenotype and are often in Backus-Naur Form (BNF). Typically, the genome is represented by a combination of 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.

$$\text{Rule} = (\text{Codon Integer Value}) mod (\#\text{ of choices}) \quad (2)$$

Using this we can introduce biases to our grammar by including multiple instances for preferred choices. For example, operand, depicted in Equation 3 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 into the designed grammar.

$$\text{<operand>:: = <choice1> | <choice1> | <choice2>} \quad (3)$$

We exploit the representational capabilities of GE resulting from the design of a grammar that defines the given search domain. GE maps the genotype to a phenotype — typically some form of program code. This phenotype can then be interpreted by the user in a predetermined manner. In these experiments, the programs created are written in a command language based on integer strings to represent sequences of MIDI notes. We design a grammar to create this command language which is in turn used to play music. An overview of the GE process including the mapping of the grammar to MIDI notes is shown in Figure 1.

4. METHOD

This section describes the methods used in composing the piano melodies that accompany this paper.

4.1 Grammar

The BNF grammar, shown below, maps the genotype (integer codon) to a series of musical events entitled notes, chords, runs, turns and arpeggios to create a musical representation. These events are re-written to numerical values that comprise a command language (series of integers) that is interpreted as individual MIDI notes.

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

<style>::=100|100|100|100|100|100|100|100
 |50,<chord>|50,<chord>|50,<chord>
 |50,<chord>|70,<turn>,100 | 80,<arp>,100
<chord>::=<int>,0,0|<int>,<int>,0
 |12,0,0|<int>,0,0|<int>,0,0|<int>,0,0
 |<int>,<int>,<int>
<turn>::=<dir>,<len>,<dir>,<len>,<stepD>
<arp>::=<dir>,<int>,<dir>,<int>,<ArpDur>

<int>::=3|4|5|7|5|5|7|7
<len>::=<step>|<step>,<step>
 |<step>,<step>,<step>
 |<step>,<step>,<step>,<step>
 |<step>,<step>,<step>
<dir>::=45|55
<step>::=1|1|1|1|2|2|2|2|2|2|2|3
<stepD>::=1|2|2|2|2|2|2|4|4|4|4|4|4
<ArpDur>::=2|2|2|4|4|4|4|4|8|8
<oct>::=3|4|4|4|4|5|5|5|5|6|6
<pitch>::=0|1|2|3|4|5|6|7|8|9|10|11
<dur>::=1|1|1|2|2|2|4|4|4|8|8|16|16|32
```

The first line creates a melody <piece> from either a single note <event> or a concatenation of note events. The inclusion of extra <event> in this first line encourages expansion of the phenotype. Each <event> starts with the indicator 111 and has the descriptors <style>, <oct>, <pitch> and <dur>. Each descriptor is mapped by the grammar in relation to what it represents. Pitch is simply a value between 0 and 11 chosen with equal probability to indicate which of the 12 pitches in the chromatic scale. Octave refers to the octave number the current event starts in and is limited to 3-6 for these experiments with a bias towards 4 and 5. Each note is assigned a specific duration ranging from a demisemiquaver (value 1) to a semibreve (value 32). As with the octave descriptor, 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 type of event determined by <style> can be a plain note denoted by 100, a chord (50), a turn (70) or an arpeggio (80). This grammar has a strong bias towards including more notes and chords as they take less time to play but can be more pivotal to the overall piece than turns. A plain note requires no further information than the octave, pitch and duration already assigned to it and so requires no further grammar. A chord (50) is defined by the pitch and duration already specified and the inclusion of either one, two or three notes played in conjunction with it.

Both turn (70) and arp (80) result in a series of notes played in sequence. 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 three. The duration of the step is limited to either semiquavers or quavers. An arpeggio is created in a similar manner.

Table 1: Attributes measured from a given individual

Name	Description
Pitch	pitch class (value 1-12)
Dur	duration
Pitch-Dur	pitch*duration
Pitch-Dist	distance between instances of a given pitch
Pitch Int	pitch interval from each note to the next
Pitch Bigram	pitch distance between successive intervals

4.2 Fitness Function

Once the grammar has mapped to the phenotype, the fitness function is called to evaluate the given individual according to a defined fitness measure. We give each individual an initial fitness based on the duration of the melody produced. We aim for a melody of duration of 300 but with a tolerance of 30. If the duration is within this tolerance the initial fitness is set to 1, else the initial fitness is calculated as the absolute value of the difference from the duration to 300 and the tolerance, plus 1. 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 Zipf's distribution of a number of attributes.

The final fitness of the individual is measured in relation to the distance in vector space of the Zipf's distribution of each of the measures shown in Table 1. These particular attributes, a subset of those used in previous experiments [15, 16], were chosen as they are most suited to the representation and methods used in this study. Measures related to the absolute pitch value were not incorporated as the grammar already controls a bias towards the use of certain octaves. Hence the term 'pitch' in these measures relates merely to the pitch class (value 1-12). Likewise we do not consider the fractal measures used in previous studies as the original duration of the pieces in these experiments is so short.

4.3 Melody Construction

The above grammar and fitness measurement create very short melodies. In order to create longer compositions we concatenate fit individuals from the final generation together. We can implement this by exploiting the fact that GE produces a population of fit individuals. In the final generation a number of the most fit individuals should be quite similar as they share common highly fit traits. Thus if we play the best individuals together we expect similar melody snippets or motifs to emerge. Previous studies in using EC for algorithmic composition have used the entire population and generations of populations in creating a single melody [18,19]. Due to the large diversity within our final population, discussed in the next section, we only con-

sider a small number of top individuals for inclusion in the final composition. A number of melodies accompany this paper displaying varying degrees of repetition and variation on a theme. Each of these longer melodies were created by concatenating the four top individuals from the final generation together.

4.4 Bass Accompaniment

Conventional piano music generally consists of two separate parts, treble and bass. Thus as an extra experiment we use a two-stage GE run that uses the best individual from the treble run to create a new grammar to compose an accompanying bass line.

Although no tonality has been enforced on the melody, the Zipf's metrics used will cause certain pitches to be played more frequently than others. Thus without predefining a key signature we can encourage a bass accompaniment to sound tonally similar to this melody by ensuring the bass exhibits the same pitch biases as the treble. We can control this effectively using our GE composition system by creating a new grammar for the bass which is derived from the evolved treble line. In this way we can ensure that only pitches already used within the piece will be considered when composing an accompaniment.

We create the grammar file for the bass part once the best individual for the treble has been found. This grammar file is created with initial predetermined lines to specify allowed note duration and octave. Only plain notes and chords are allowed in the bass grammar. The line of the grammar that defines the allowed pitches is created from an ordered list of pitches in the treble line. From this we create a list of pitches available to the bass that includes the top notes from the melody four times, the next two notes three times, the following two notes twice and includes the sixth, seventh and eight most frequent note once. This creates a bias within the bass towards the pitches most frequently used within the melody. For example if the most frequent pitches in the treble melody were A, C, F#, G, D, E, D# and C#, in decreasing frequency, the line:

```
<pitch>::=9|9|9|9|0|0|0|6|6|6|7|7|2|2|4|3|1
```

would be added to the predefined grammar, completing the grammar for the GE to evolve the bass accompaniment.

The bass grammar considers the tonality of the treble and bass parts but it does not take into account the progression or timing between the two. A simple method to create an accompanying line is to create one bass part that repeats underneath all four similar melodies. To achieve this we must be more strict in the duration of the treble melody evolved; if we want the accompanying bass to repeat twice under each melody individual we must ensure each bass individual is exactly half the duration of the treble. Thus we re-run the experiment again with a target duration of 128 for the treble, 64 for the bass and zero tolerance for both parts. As a duration of 1 represents a demisemiquaver, a duration of 32 could represent one bar in 4/4 time. Hence we can consider the melody to be of length four bars and the bass to be of length two, although the duration of individual bar lengths is not enforced.

To measure the fitness of the bass individual we again consider a Zipf's distribution, but this time on the harmonic relationship between the pitches in this bass and the melody it is accompanying. To consider the harmonic progression between the two parts we must examine the relationship between each pair of notes at every time-step. To examine this we expand out the pitch line for both the treble and bass so that we have a value at each instance (each moment of duration 1). For example, if there is a crotchet (duration 8) played at D (pitch 2) we represent this with a list of 8 values of 2. This results in two lists, one for treble and one for bass that indicate the pitch of each line at every moment of duration. In the case of a chord, only the root note of the chord is considered. We then subtract the bass from the treble list to create a list of intervals. As we are only considering pitch values within an octave, this results in a negative value should the pitch value of the bass be higher than that of the treble. We correct this by adding a value of 12 when this occurs.

We then categorise the resultant interval list into a list of rankings according to standard Western tonality. These rankings indicate how consonant or dissonant an interval is, with 0 being the most consonant (least dissonant) and 12 being the most dissonant (least consonant). From this list of rankings (which is already sorted), we can then enforce a Zipf's distribution and adjust the fitness in accordance to the deviation from this distribution as per the treble part.

The system described is implemented in python using PonyGE `https://code.google.com/p/ponyge/`. Details of the experiments run are given in the following section.

5. COMPOSITIONS

The experiments were run with a population of 200 for 50 generations. 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. Each experiment was run with a minimising fitness function whereby zero is the ideal fitness.

A selection of melodies created by this system are available at `http://ncra.ucd.ie/Site/loughranr/smc_2015.html`. In this section we discuss the creation of the melodies in relation to fitness evolution, the Zipf's distributions, variety in the final generation and the creation of an accompanying bass part.

5.1 Short Melodies

5.1.1 Fitness Evolution

The progress of any evolutionary run is best examined by observing the progress of the average and best solution in successive generations across multiple runs. Figure 2 displays the average versus best fitness across 30 evolutionary runs for the creation of the melody line. We note that the fitness is calculated directly but the natural log is shown for illustrative purposes. It is evident from this graph that on average a near optimal fitness can be found after about 30 generations. In contrast to this, the average fitness remains

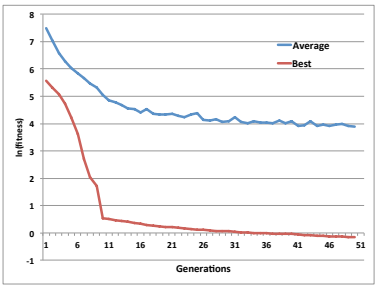

Figure 2: Average vs. Best Fitness over 50 generations average over 30 runs.

Table 2: Zipfs measurements from individual melody attributes. Numbers in parenthesis indicate the ideal values.

Attribute	Slope (-1)	R^2(1)	Fit (0)
Pitch	-1.01	0.94	0.07
Duration	-1.1	0.98	0.13
PitchDur	-1.0	0.95	0.06
Interval	-1.0	0.93	0.07
PitchDist	-1.0	0.91	0.08
Bigram	-1.02	0.95	0.08

quite high. This implies that after 50 generations the population is still very diverse. We consider the reason for this diversity later, but first we examine an individual melody in terms of the fitness attributes measured.

ShortMelody is the best evolved individual across all runs with a final fitness of 0.49. This melody contains all melodic events the grammar is capable of producing — single notes, chords, turns, runs and arpeggios. The evolution of each of the attributes is shown in Figure 3. These plots show the progression of the slope and R^2 for each of the six measured attributes for the best and median individual in each generation, measured by fitness. As expected the best value approaches the ideal for each value quickly whereas the median values show much more variation. It should be noted that the median value at generation 1 tends to be zero. This is because the first generation have many very weak individuals (more than half the population) that are very short resulting that the median's initial fitness is too weak to be adjusted using Zipf measurements. This variety with the median values across the generations show that while the best fitness is easily met, the population remains diverse in relation to each of the fitness attributes.

5.1.2 Zipf's Distribution

Figure 4 displays the distribution of each of the six attributes measured from ShortMelody in relation to the Zipf's ideal. These plots clearly illustrate that the points converge to a straight line with a negative slope. For a closer inspection Table 2 displays the the specific values from these plots for slope, R^2 and attribute fitness. Despite the small number of points on these graphs, they do indicate that the attributes display Zipf-like distribution with each of their slopes approaching the ideal of -1 and R^2 approximating the ideal of 1.

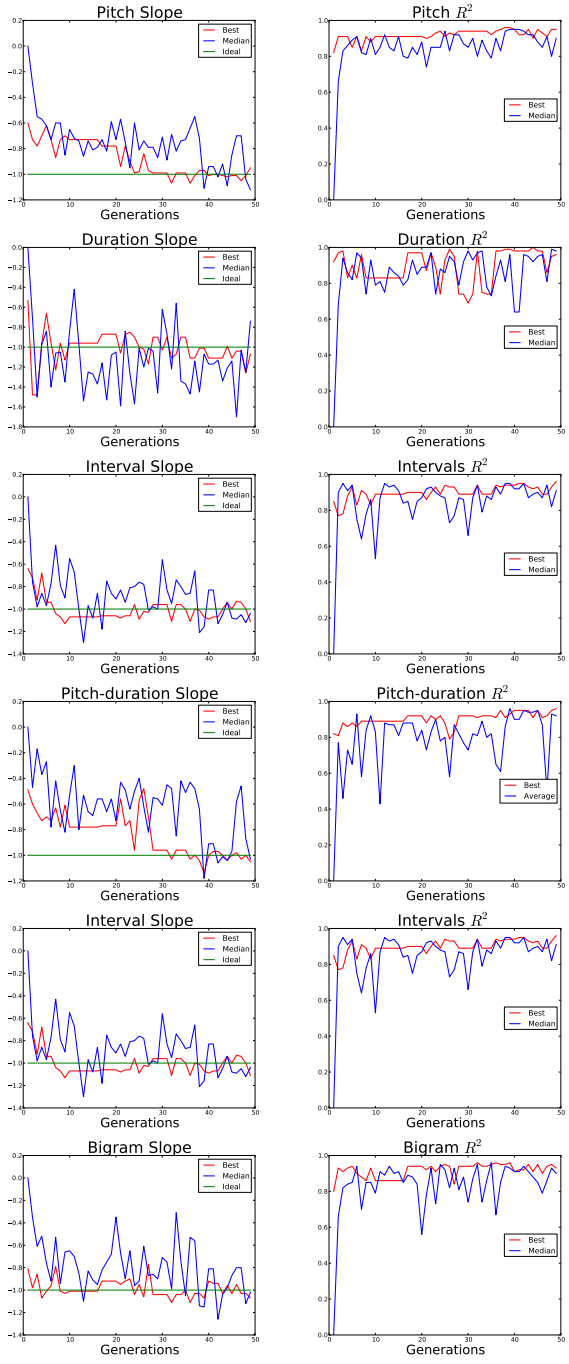

Figure 3: Evolution of attributes for most fit melody

5.1.3 Final Generation

Figure 2 shows that after 50 generations there is still a large difference between the average and best fitness in a population, indicating that the final population is still very diverse. To determine why this may be, we examine the individuals within the final population of an evolutionary run. Figure 5 shows the fitness values within the final population. These show that over 50% of the population do have very low (good) fitness. The distance between the average and the best is caused by a small number of very weak individuals that drag the average up. Examining the lengths in the final population indicate that a similar number of individuals have very short durations. As the initial fitness is

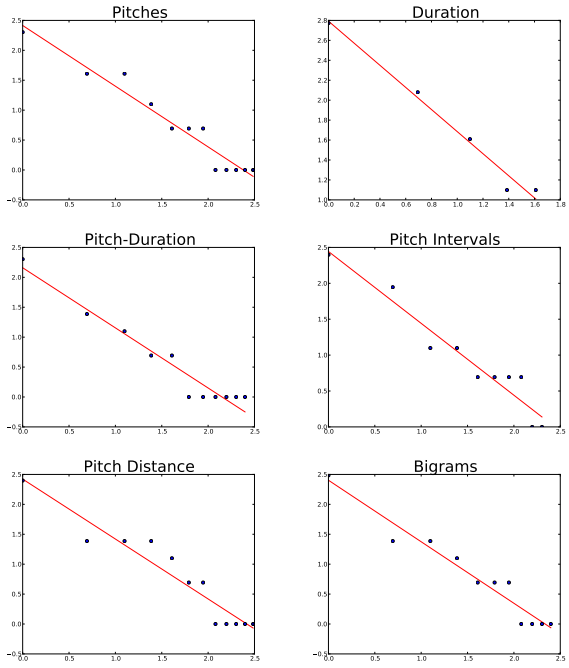

Figure 4: Attribute distributions for most fit melody. The red line indicates the Zipf's ideal distributions.

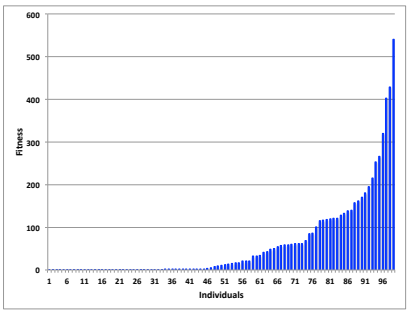

Figure 5: Overall Fitness Values in final generation.

based on length, a short duration will dramatically increase the fitness, thus a small number of short melodies will alter the average fitness of that generation. Similarly, we can examine individual attribute fitness measures within the final generation. Again we find that the attributes approach ideal values for those individuals with low fitness but deviate from the ideal in weaker individuals in the population. The attribute that shows most deviation is duration. This is unsurprising as the addition or removal of a single turn can significantly alter the instances of a given duration. Even so, Figure 5 clearly shows that there are a large number of melodies with a very good fitness, hence we can be confident in our choice of the single best or make use of a combination of the best as described in the following section.

5.2 Composite Melodies

Eight composite melodies accompany this paper displaying varying degrees of repetition and variation on a theme. Each of these longer melodies were created by concatenating the four top individuals from the final generation together. Melody4.mp3 offers an interesting motif emerging within the middle of each individual. This motif is notated in Figure 6. Similar themes can be heard to emerge in the other melodies. These motifs or themes ground the compositions giving them a sense of oneness and modularity. The emergent themes can vary in length; Melody6.mp3 can be heard to root itself in a long F# both at the middle and end of each individual giving a very repetitive flow to the melody. Each melody presented displays the events produced by the grammar in terms of runs, chords and arpeggios, they all display some level of modularity through repetition of motifs but they are all very distinct from one another. Although Melody1, Melody2 and Melody3 result in the best fitness, the authors found Melody4 and Melody5 to be more pleasing to the ear. This raises questions as to how much merit we should attach to fitness measures such as these — the fitness function can be used to traverse the search space but it did not necessarily lead to the 'best' melody.

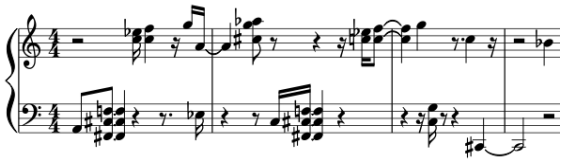

Figure 6: Theme emergent in Melody4.mp3

5.3 Bass Accompaniment

Although some of the melodies, such as that illustrated in Figure 6 are written on both staves, it is a single part melody that is evolved. We ran the experiment again to create both treble and bass parts producing three compositions as described in Section 4.4. In each of these compositions we can hear a repetitive bass part underlying the melody. In Accompany1 and Accompany2 these do not quite fall in time with the upper line, but the fact that they are of strict durations (bass 64, treble 128) keeps the two parts together in a cyclical manner. Accompany3 offers a much more syncopated accompaniment that compliments the treble melody more pleasantly.

A notable, if somewhat obvious, point to make is that it is much more difficult to compose two accompanying lines. One method around this would be to co-evolve the two parts together, although we find something unnatural about this. While there are exceptions where two melodies are composed together, in general when we think of an accompanying line, it is just that: a new part that is written to complement another already composed melody. We have avoided specifying key or time signatures through these experiments, instead opting for ranking and statistical measures to control the content. We feel that this may work well between lines in regards to pitch, as we can constantly measure the harmonic distance between two accompanying parts, but the temporal nature of music gives rise to difficulty when considering rhythm. The repeating measure reported here serves it's function but we acknowledge that it is very limiting. In future work we hope to inves-

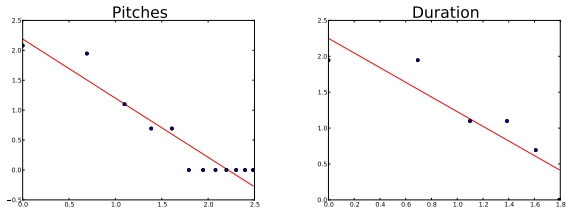

Figure 7: Distribution of the Pitch and Duration attributes in a melody created with a simple Grammar

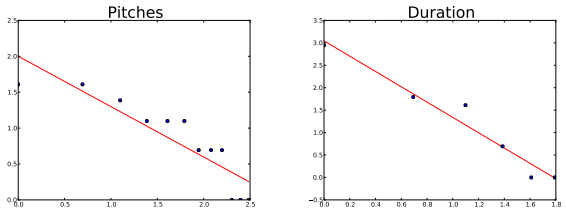

Figure 8: Distribution of the Pitch and Duration attributes in a melody created with a simple Fitness

tigate better methods of creating melodies that temporally and rhythmically complement one another.

5.4 Compositional Elements

The compositions created by this system are largely due to three distinct processes:

1. Representation created by the grammar
2. The Duration and Zipf's Law Fitness Function
3. Repetition of motifs from concatenation of individuals

To determine the significance of each of these aspects, we reran the experiment with varying combinations of each aspect and examined the output. BasicGram1 and BasicGram2 are created using a grammar that only allowed single notes. BasicFit1 and BasicFit2 are evolved with a fitness function that was targeted solely on the length of the melody, disregarding any Zipf-based measurements. ShortMelody is the single best individual evolved, but is not concatenated with any other individuals from the population.

5.4.1 BasicGram

From listening to the melodies created using the basic grammar, it is clear that these are less interesting, less engaging and less pleasant than those created with the more involved grammar. Nevertheless, these BasicGram melodies have equally good (or even better) fitness as our other composite melodies according to our defined fitness function. Figure 7 displays the Zipf's distributions for the pitch and duration attributes for BasicGram1. As expected, these display typical Zipf's distributions with slopes of -0.99 and -1.02. This demonstrates that we need more than a good statistical fitness measure to create a good melody.

5.4.2 BasicFit

BasicFit1 and BasicFit2 were evolved using the full grammar but with a minimal fitness function that only measured the duration of the piece. Thus the best fitness of 1 was reached very quickly, within five generations. Although they were not taken into account during evolution, we calculated the Zipf's distribution for each attribute used in the rest of the experiments. A plot of distribution for the pitch and duration are shown in Figure 8. Although these may initially appear to portray a Zipf-like distribution, a closer analysis shows that the slope for the Pitch and Duration attributes are -0.7 and -1.7 respectively. Similarly, the slopes of the distributions for the pitch-duration, intervals, pitch-distance and bigram attributes were -0.52, -0.58, -0.89 and -0.81. Nevertheless, from listening to these melodies, we find them to be more interesting than those evolved using just the basic grammar.

5.4.3 Short Melody

As discussed in the fitness results, ShortMelody is the best individual found throughout our evolutionary run. It displays all of the compositional elements from the grammar – notes, turns and chords — and it exhibits very accurate Zipf's distribution on each of the measured attributes. Melody1 is the concatenation of this melody with the next top three individuals from the final generation of that run. From listening to both it is evident that the longer concatenated melody is more pleasing and offers more structure than the original short melody on it's own. This aspect of emergent motifs due to repetition is even more evident in other compositions such as Melody4. The degree of repetition is related to the similarity between the selected individuals. In some final generations the top individuals are all very similar giving a high degree of repetition and musical motifs. In other experiments, one or more of the top individuals differ significantly yet have similar fitness. As the concatenation of individuals is one of the most effective methods for creating musicality with this system, we plan to explore this process further with a view to using a similarity measure between individuals as a means of concatenating them into one composition.

Overall, we found that the grammar and representation used in these experimented in combination with the concatenation of a number of highly fit individuals have had a more pleasing aesthetic result in the creation of musical compositions than the use of the Zipf's based fitness. We encourage the reader to evaluate these for themselves, but the authors concurred that in regards to musicality, the melodies created using the full system [1] are much more pleasant to the ear than those from a single individual or those that do not make use of the full grammar. Evolution is driven by the fitness function used, so it is our conclusion that future work should be focussed on finding a more beneficial and musical way of measuring this fitness.

6. CONCLUSIONS

We have composed a series of piano compositions with Grammatical Evolution driven by a Zipf's Distribution of

[1] in particular we enjoyed Melody4, Melody5 and Accompany3

a variety of compositional attributes. A notable issue with the compositions produced is that they lack overall form. We would like to continue this work to develop the progression of the pieces to include a distinctive beginning, middle and end and ideally follow some discernible trajectory as the piece develops. We plan to develope future versions of this system with a better fitness function. Although Zipf's distribution of the attributes measured have been shown in previous literature to correlate with musical pleasantness, we did not find them to be the most important aspect in creating an interesting composition. Instead we found that exploiting GE's use of grammar and concatenating similar but not identical individuals together was more important in the musicality of the result.

While Section 5 offers details and results showing the workings of the experiments run, the best measure of a compositional system is in judging it's musical output. Inherently, this is an aesthetic judgement and it is one that is not easily defined or quantified. Nevertheless, the authors find merit in the compositions produced. We acknowledge that there is a lack of form to the compositions, and that there is notable room for improvement in using the system to create two part melodies. However, as a new system incorporating GE with new grammars and representation it offers worth as a compositional aid; it can create original musical ideas that could be utilised and modified by a human musician in the creation of a larger composition.

7. ACKNOWLEDGMENTS

This work is part of the App'Ed (Applications of Evolutionary Design) project funded by Science Foundation Ireland under grant 13/IA/1850.

8. REFERENCES

[1] M. O'Neill and C. Ryan, *Grammatical evolution*. Springer, 2003.

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

[3] 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.

[4] 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.

[5] D. Perez, M. Nicolau, M. O'Neill, and A. Brabazon, "Reactiveness and navigation in computer games: Different needs, different approaches," in *Computational Intelligence and Games (CIG), 2011 IEEE Conference on*. IEEE, 2011, pp. 273–280.

[6] M. O'Neill and A. Brabazon, "Evolving a logo design using lindenmayer systems," in *IEEE World Congress on Computational Intelligence*. IEEE, 2008, pp. 3788–3794.

[7] 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.

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

[9] 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.

[10] 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.

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

[12] A. O. de la Puente, R. S. Alfonso, and M. A. Moreno, "Automatic composition of music by means of grammatical evolution," in *ACM SIGAPL APL Quote Quad*, vol. 32, no. 4. ACM, 2002, pp. 148–155.

[13] 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.

[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] 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.

[16] B. Manaris, D. Vaughan, C. Wagner, J. Romero, and R. B. Davis, "Evolutionary music and the Zipf-Mandelbrot law: Developing fitness functions for pleasant music," in *Applications of Evolutionary Computing*. Springer, 2003, pp. 522–534.

[17] G. K. Zipf, *Human behavior and the principle of least effort*. Addison-Wesley Press, 1949.

[18] R. Waschka II, "Composing with genetic algorithms: GenDash," in *Evolutionary Computer Music*. Springer, 2007, pp. 117–136.

[19] A. Eigenfeldt and P. Pasquier, "Populations of populations: composing with multiple evolutionary algorithms," in *Evolutionary and Biologically Inspired Music, Sound, Art and Design*. Springer, 2012, pp. 72–83.