Clustering Agents for the Evolution of Autonomous Musical Fitness

Róisín Loughran(✉) and Michael O'Neill

Natural Computing Research and Applications Group, University College Dublin, Dublin, Ireland
roisin.loughran@ucd.ie

Abstract. This paper presents a cyclical system that generates autonomous fitness functions or *Agents* for evolving short melodies. A grammar is employed to create a corpus of melodies, each of which is composed of a number of segments. A population of Agents are evolved to give numerical judgements on the melodies based on the spacing of these segments. The fitness of an individual Agent is calculated in relation to its clustering of the melodies and how much this clustering correlates with the clustering of the entire Agent population. A preparatory run is used to evolve Agents using 30 melodies of known 'clustering'. The full run uses these Agents as the initial population in evolving a new best Agent on a separate corpus of melodies of random distance measures. This evolved Agent is then used in combination with the original melody grammar to create a new melody which replaces one of those from the initial random corpus. This results in a complex adaptive system creating new melodies without any human input after initialisation. This paper describes the behaviour of each phase in the system and presents a number of melodies created by the system.

Keywords: Algorithmic composition · Grammatical Evolution · Clustering · Self-adaptive system · Autonomous fitness function

1 Introduction

Boden has suggested three ways in which computers can display creativity [2]:

– Combining novel ideas
– Exploring the limits of conceptual space
– Transforming established ideas that enable the emergence of unknown ideas

Grammar based evolutionary methods such as Grammatical Evolution (GE) [4] offer an interesting parallel to such processes. The 'combination of ideas' concept can be likened to the crossover operator used in evolutionary systems, while 'exploration' can be likened to the mutation operator. The use of grammars in GE can facilitate the third idea of 'transformation' listed above. Thus we propose that grammar-based evolutionary systems are well-suited to creative tasks such as melody writing. The creation of melodies offers a particularly difficult evolutionary computational challenge as there is no absolute correct answer; judging whether one melody is better than another is inherently a subjective matter. Often, this problem is addressed by using a human as a fitness function, using a set of known musical rules or comparing the music to a given style or genre. Each of these methods are based on the assumption that human-made music is best — and consequently is what is being searched for. But there already is an abundance of music being created (by humans) that follow such rules, with more being created every day. In looking at algorithmic composition as a computational problem, we are given an opportunity to consider it from a different angle. Assuming that the music created by machines must automatically be judged in human terms is an assumption that has the potential to limit the capabilities of any computationally creative system [16]. As Boden stated [3]:

> *'The ultimate vindication of AI-creativity would be a program that generated novel ideas which initially perplexed or even repelled us, but which was able to persuade us that they were indeed valuable'.*

As long as autonomous creative systems are focussed on human judgement, this will be impossible to realise. For these reasons this paper proposes a system that evolves melodies, not according to any musically-derived fitness function, but by developing an autonomous fitness function that is created in response to the system itself.

The proposed system creates a population of *Agents* that result in a numerical output for a melody, evolved using a given melody corpus. An Agents' fitness is not based on any musical quality but on how well each individual Agent clusters the melody corpus in relation to the average clustering of the entire population of Agents. As such, it mimics the social phenomenon of agreement: those Agents that conform to the population are given better fitness than those that do not. In this way an individual Agent does not have any merit on its own — its performance can only be measured in relation to the overall behaviour of the population. Once an Agent is evolved, it is used in a further evolutionary run as a fitness function to create a new melody. This new melody replaces one of those in the original corpus and the process is repeated. This results in a cyclical process of Agents used to create melodies that are in turn used to create new Agents. Thus we present a complex adaptive system that continuously updates in response to its own behaviour without any human influence.

The following section describes relevant literature in evolutionary systems applied to melodic composition. Section 3 describes each stage of the system. Experimental results and the behaviour of each stage of the system is described in Sect. 4. Conclusions and proposed future work are given in Sect. 5.

2 Previous Work

This study proposes a novel method for using GE to evolve melodies, focussing on developing an autonomous fitness function that has no a priori musical knowledge or preference. In recent years, a number of EC methods have been applied to the problem of algorithmic composition. Genetic Algorithms (GA) have been applied in the systems GenJam to evolve real-time jazz solos [1], GenNotator to manipulate musical compositions using a hierarchical grammar [23] and to create four-part harmony from music theory [9]. More recently, adapted GAs have been used with local search methods to investigate human virtuosity in composing with unfigured bass [18], with a grammar to augment live coding in creating music with Tidal [10], and with non-dominated sorting in a multi-component generative music system that could generate chords, melodies and an accompaniment with two feasible-infeasible populations [20]. Genetic Programming (GP) has been used to recursively describe 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 [7]. Interactive Grammatical Evolution (GE) has been used for musical composition with promising results [21]. GE has also been used recently with autonomous fitness functions based on statistical measures of tonality and the Zipf's distribution of musical attributes [13,14]. These studies found the musical representation created by the grammar and the combination of individuals from the final population could be as important as the fitness function. Some studies have addressed the problematic issue of determining musical subjective fitness by removing it from the evolutionary process entirely. GenDash was an early developed autonomous composition system that used random selection to drive the evolution [25]. Others used only highly fit individuals within the population from initialisation and then used the whole population to create melodies [1,8].

The evolution of a population of individual Agents (or 'Critics' or similar terminology) that adjudicate a melody in some way has been proposed in a number of notable studies. The concept of populations co-evolving in a composer-critic paradigm was presented in [24]. This modelled the production of birdsong in nature by co-evolving males who composed songs along with female critics who decided, based on these songs, who to choose as a mate for the next generation. An evaluation framework consisting of a number of critics was proposed in [19]. This study induced a set of critics from a set of musical examples after first specifying specific musical criteria. The system then created music and was evaluated by a set of human listeners. A distributed population of autonomous composing agents is described in [17], which co-evolved agents with repertoires of melodies according to a measured 'sociability'. This sociability was measured in terms of similarity of the agent's repertoires; individual melodies survived or were altered depending on reinforcement feedback between co-evolving agents. This study differs from the proposed method as it is the correlation of a individual Agent's clustering of melodies to that of the (single) population that is measured in this system rather than a direct similarity measure between melodies.

A notable study demonstrated that in Computationally Creative Evolutionary systems, it is only important that the decision of fitness need be defensible; what makes one creative item better than another may not be what a human would choose but it must be a sensible, defensible and reproducible choice by the computer program. In other words there must be a logical and explainable method in assigning fitness measures. This was investigated using the idea of a preference function by measuring qualities such as specificity, transivity and reflexivity to determine the choice of a system in a number of subjective tasks [6]. Such a measure may not agree with what a human may choose as the best but, most importantly, it agrees with itself. This preference function chooses one item over another due to a logical system of comparing between items and determining a decisive preference. We try to build on this idea in the system proposed. Creating a fitness function (or Agent) based on a dynamic measurement of the system rather than a typical human measure is a key idea in the proposed system.

It has been proposed that using a pre-specified objective is not necessarily the best approach to searching. This theory suggests that searching for novelty is a better method when considering a problem, that good solutions can be found when looking for a different solution or when searching for no particular solution at all [12,22]. Such a theory fits very well in searching any creative space. A musician may not know exactly what piece of music they are trying to create when they start, they work through ideas, changing their process and hence their output as they observe what they are creating. Furthermore it was discussed in [16] that using human-based fitness measures may not be ideal in generative music; the prevalent and consistent adjudication of autonomously generated music against human opinion or measures determined from human-created music theory may in fact be limiting the potential of systems that could create music outside of such constraints. It is for such reasons that the current system develops an autonomous fitness function that is based purely on self-referential organisation of melodies by the system as it develops; the proposed method constitutes a complex, adaptive system that recursively amends an initialised corpus of melodies in response to the continually updating fitness measure. The steps in this process are detailed in the following section.

3 Method

The main focus of the system is in the creation of a fitness function that can be used to search through a population of melodies and 'adjudicate' them by means other than using known musical qualities. To do this a population of fitness functions is evolved, which from henceforth will be referred to as *Agents* in this work. Each Agent is used to cluster a population of constructed melodies. An overall clustering measure from the Agent population is calculated and each individual Agents' fitness is measured in relation to how much it agrees with this general clustering. The form of the Agents is specific to the melodies created for these experiments; Agents are constructed as a linear combination of the distances between each segment of the given melodies. The full proposed system is cyclical: a corpus of melodies is used in the evolution of an Agent which is used to evolve a new melody to be included in the original corpus, and the cycle repeats. This section describes the representation of the melodies, the representation of the Agents and how they are used together as the system evolves.

3.1 Melody Representation

Each Agent is evolved according to its correlation with the population in the clustering of a selection of melodies. Throughout the system, melodies are created using a previously developed system for composing short melodies with GE. A full description of this system and the results obtained can be found in [15]. The grammar used is based on:

```
<piece>::= <seg><seg><seg><seg><seg><seg>
<seg>::= <event><event><event><event><event>
<event>::= <style>,<oct>,<pitch>,<dur>
<style>::= <n>|<n>|<n>|<n>|<n>|<n>|<chd>|<chd>|<chd>|<chd>|<turn>|<arp>
<chd>::= <in>,0,0|<in>,<in>,0|<in>,<in>,<in>
<turn>::= <dir>,<len>,<dir>,<len>,<stp>
<in> ::= 3|4|5|7|5|5|7|7
<len>::= <stp>|<stp>,<stp>|<stp>,<stp>,<stp>|<stp>,<stp>,<stp>,<stp>
<dir>::= down|up
<stp>::= 1|1|1|1|1|2|2|2|2|2|2|2|2|3
<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
```

This grammar creates a melody `<piece>` containing six segments, each comprising of a number of musical events. Each `<event>` can either be a single note (`<n>`), a chord, a turn or an arpeggio. A single note is described by a given pitch, duration and octave value. A chord is given these values but also either one, two or three notes played above the given note at specified intervals. A turn results in a series of notes proceeding in the direction up or down or a combination of both. Each step in a turn is limited to either one, two or three semitones. An arpeggio is similar to a turn except it allows larger intervals and longer durations. The application of this grammar results in a series of notes each with a given pitch and duration. The inclusion of turns and arpeggios allows a variation in the number of notes played, depending on the production rules chosen by the grammar.

The use of this grammar results in MIDI melodies split into six segments. A pitch vector for each segment is found by expanding the segment to give the pitch value at each demisemiquaver. This vector is normalised by setting the first value to zero (and transposing the remaining pitches accordingly) resulting in a pitch contour for each melodic segment. These contours can then be compared directly. In this manner, a distance can be measured between each of the segments of a given melody. The distances between all six segments in a melody can be represented numerically in a 6 by 6 matrix. This matrix is symmetrical about the diagonal, allowing it to be collapsed into a single vector of length 15 (i.e. $5+4+3+2+1$) that represents the distance between each pair of segments. It is from these distances that an Agent measures a given melody.

3.2 Agent Construction

As described above, each melody may be represented by a vector of 15 distance values. Each Agent is formed to syntactically result in a linear combination of these 15 measured distances. This is realised using GE with the following grammar:

```
<expr> ::= <O><D1><O><D2><O><D3><O><D4><O><D5><O><D6><O><D7><O><D8>
           <O><D9><O><D10><O><D11><O><D12><O><D13><O><D14><O><D15>
<O> ::= <op><scalar>
<op> ::= + | - | *
<scalar> ::= 1 | 2 | 3 | 4 | 5
```

This very simple grammar takes a linear combination of each of the 15 distance measures used to represent a given melody. Thus any Agent will output a single numerical value for each melody. The fitness function used to evolve a given Agent is based on a measure of how the individual Agent clusters the melodies in relation to how the current population of Agents cluster the melodies. Thus the Agent has no merit from its own output, but only in relation to the way in which it performs in respect to the rest of the population. This idea forms the crux of the proposed system. It can be summarised in the following steps:

- Each Agent clusters the corpus of melodies
- The most prevalent clustering of all melodies across all Agents is noted
- Each Agent is assigned fitness according to how well it correlates or ‘agrees’ with the overall clustering

As each Agent produces a numerical result for each melody, clustering in one dimension is only to be considered. This was implemented using Jenks natural breaks optimisation technique [11]. This method determines the best division of data between classes by seeking to minimise the average standard deviation of each element from the class mean while maximising each class’ mean from those of the other classes. We consider between two and six clusters and choose the option with the best return of variance. To acquire a measure of classification we consider whether or not each pair of the melodies are found to be in the same class, assigning a value of 1 if they are and 0 otherwise. This results in a matrix of 1s and 0s again symmetrical about the diagonal (if melody 2 is in the same class as melody 3 then the reciprocal is also true). As an illustrative example, consider the arrangement [a, a, b, b, c, c] — a corpus of six individuals that should be logically clustered into three groups of two elements. If these values are indexed 1–6, all possible pairings between these six individuals can be represented by the 15-length vector:

[(1,2),(1,3),(1,4),(1,5),(1,6),(2,3),(2,4),(2,5),(2,6),(3,4),(3,5),(3,6),(4,5),(4,6),(5,6)]

If we consider ‘1’ indicates both indexed elements are in the same cluster and ‘0’ implies not, the clustering of the 6-element list of letters should be:

[1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1]

A similar vector can be calculated for any known clustering of one-dimensional data — such as that produced by each Agent on a corpus of melodies.

A population of 100 Agents will thus result in 100 binary-valued vectors. These are summed across all 100 Agents and normalised within the limits [0, 1] to give the *Population Clustering*. The mean squared error of the 'cluster result' of the given Agent to this Population Clustering is calculated and assigned as the fitness value of that Agent. By minimising this fitness the Agent that conforms with the majority clustering of the population is assigned best fitness in the evolutionary run. The experiments in Sect. 4 describes two implementations of this phase of the system — the preparatory run and the full system implementation. In the preparatory run this Population Clustering is replaced by an ideal clustering known by creating melodies of three specific shapes; in the full evolutionary system it is updated with each new Agent population at each generation.

3.3 Evolving a Melody

Once a best Agent has been evolved, this can then be used to evolve a new melody. GE is employed with the grammar described above in Sect. 3.1 and this best Agent as a fitness function to evolve a new melody. Each Agent is a linear combination of the distances measured from the six segments within the given melody. Hence the fitness of each melody is calculated as:

$$\text{fitness}_{melody} = abs(\text{Agent}(\text{distances}_{melody}))$$

Minimising this fitness will result in the melody with the lowest absolute output, as measured by the Agent, being deemed the best melody. This melody replaces one of the randomly generated melodic representations. After 30 full cycles of the system, the corpus has been re-populated with melodies created by the system.

3.4 Full System

As detailed in the following section, an preparatory phase is used with melodies of known shape to create an initial population of Agents. Once this is completed, a full run of the system can be undertaken using a randomly initialised corpus of melodies. The saved Agents from the preparatory phase are used as the initial population in evolving a best Agent, which is then used to evolve a best melody, which is subsequently used to update the corpus. A graphical overview of the system is shown in Fig. 1.

The phases in these experiments are all based on evolutionary runs. The parameters shown in Table 1 (chosen as those typical in the literature) are common to all experiments; generation and population size can vary and are stipulated in each relevant section.

4 Experimental Results

This section describes the behaviour of each individual section of the system. A selection of melodies produced and described below can be found at http://ncra.ucd.ie/Site/loughranr/evo_2017.html.

Table 1. EC parameters common to each evolutionary phase.

Parameter	Value
Selection	Tournament (size 2)
Crossover rate	0.7
Mutation rate	0.01
Initial genome length	100
Elite size	1

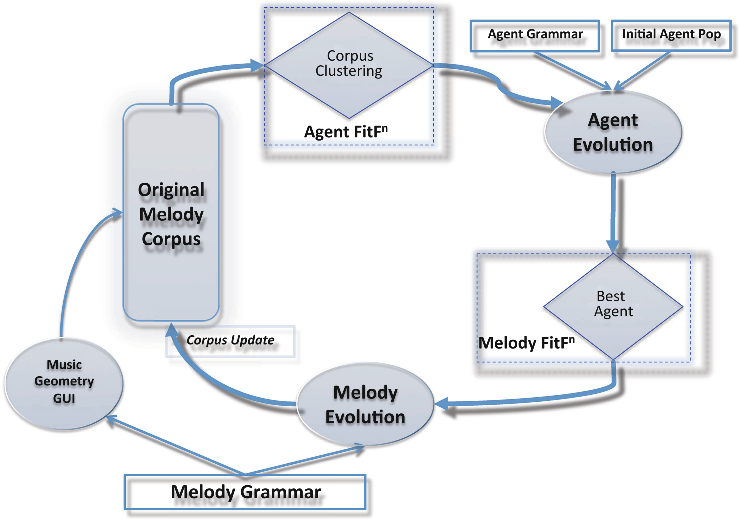

Fig. 1. Graphical overview of the full proposed system.

4.1 Preparatory Cycle

This proposed system is cyclical and self-referential, each stage of the cycle responding to the current state of the system. To start the system, however, we initialised a population of melodies that conformed to three pre-defined shapes, thus having a predictable clustering. This initial melody corpus was used to create an initial population of Agents, which can be shown to possess the required (or typical) clustering ability. These initialisation steps are described below.

Initial Melody Corpus. To create the initial population of Agents, a preparatory training corpus of 30 melodies conforming to the three shapes in Fig. 2 was created. The melodies were created to contain six segments, using the grammar described above in Sect. 3.1. Each of the three shapes can be represented by measuring the distance between each of the six given points of the shape, again resulting in a 15-point vector. Melodies can then be evolved towards one of these shapes using the mean-squared error between the melodic contour representation and that specified by the graphical shape. The evolutionary strategy used to evolve the melodies was based on a population-based hill-climbing strategy entitled the *Music Geometry GUI*, described in full in [15]. This strategy applied variable neighbourhood search as a series of operators of increasing complexity until an improvement was found. Using this method, an initial corpus of 30 melodies was evolved containing 10 Hexagonal, 10 Returning and 10 Alternating shapes.

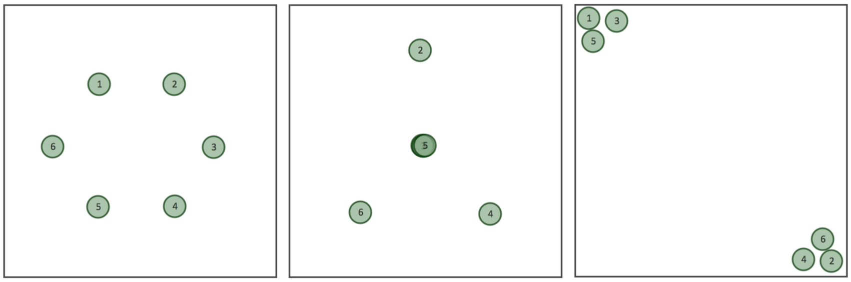

Fig. 2. Shape targets for the Hexagonal, Return and Alternating melodies

Initial Agent Population. The 30 melodies in the given corpus are clustered using natural breaks as described in Sect. 3.2. The initial melody corpus described above contains melodies that form three distinct clusters — Hexagonal, Returning and Alternating melodies. As this is known, we can create a target 'ideal' clustering 435-length vector (30×30 matrix again reduced along the diagonal) for the evolving Agents. The fitness measure of each Agent is calculated as the mean-squared error from its clustering vector to this ideal target clustering vector. This was run independently 100 times with a population of 100 over 50 generations. A plot of the population average and best fitnesses achieved over these 100 runs is shown in Fig. 3. This shows a typical evolutionary response with a steady reduction in both average and best fitness over successive generations. Each Agent consists of a linear combination of each of the 15 distance measures in each melody e.g.

```
-1D1+3D2*4D3-0D4+2D5-2D6+5D7*4D8*4D9*1D10+4D11+5D12-5D13-3D14-1D15.
```

These 100 best evolved Agents were examined and it was found that, although they achieve similar fitness results, no two were syntactically identical. Hence we can be confident there is enough diversity among these Agents to use them as an initial population for an evolutionary run. Evolving 100 unique solutions to a problem indicates that it is a simple problem with many local optimal solutions. Such a challenge is ideal for initialising a problem such as this: the aim is to create a diverse group of individuals that have some ability relevant to the domain, yet are not specialised.

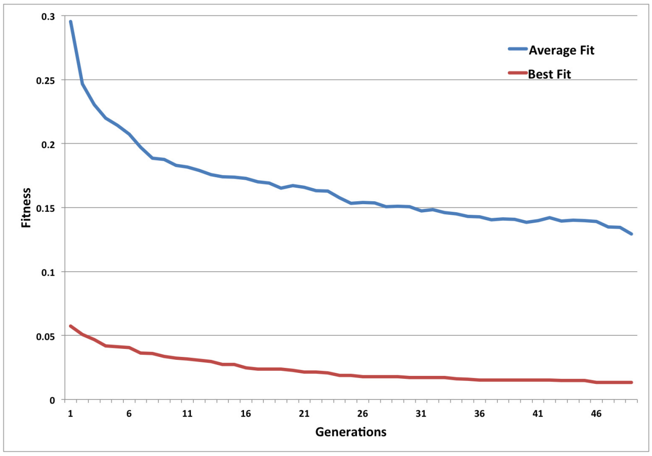

Fig. 3. Average vs. best fitness of the evolution of the initial Agents averaged over 100 runs.

Of the 100 best Agents it was found that 30 achieved a perfect fitness score of 0. Further to such fitness measures, it is possible to examine the actual clustering of these Agents on the data. Using the original training (ordered as 10 Hexagonal, 10 Alternating, 10 Returning) the ideal clustering order (of the indexes of the melodies) is naturally:
[(0,1,2,3,4,5,6,7,8,9) (10,11,12,13,14,15,16,17,18,19) (20,21,22,23,24,25,26,27,28,29)]
In comparison to this ideal cluster pattern, 83 melodies were clustered otherwise by the best Agents. Considering 30 melodies over 100 experiments, this results in a Clustering Accuracy of 97.23%. As a test to the generality of these Agents, the data corpus was rearranged to be ordered [Hexagonal, Alternating, Returning, Hexagonal, Alternating, Returning...] resulting in ideal clustering of
[(0,3,6,9,12,15,18,21,24,27) (1,4,7,10,13,16,19,22,25,28) (2,5,8,11,14,17,20,23,26,29)]
When compared it was found that 83 (different) melodies were mis-clustered by the Agents, again leading to a Clustering Accuracy of 97.23%. This indicates that this initial population of Agents can generalise and are robust to re-arranging the melody corpus. These Agents were then used to cluster unseen data in a full run of the system.

4.2 Full Cycle

Each full cycle of the system evolves one best Agent, which is used to create one best melody, that replaces one of the melodies from the original corpus. Hence in each cycle the corpus is different only by one melody, and the evolving population of Agents is initialised *each time* from the population created in the preparatory run. The Agents are evolved according to an agreement among the population as to what way to cluster the current corpus of melodies — hence the system learns through self-organisation.

The known shapes for melodies shown above in Fig. 2 were chosen to create an initial population of Agents to seed the Agent evolution at the beginning of a full experiment. This is to ensure the initial population of Agents contains some useful ability — evolving towards the clustering consensus of randomly generated Agents does not necessarily hold any merit. Once an Agent population has been created that can reliably classify the ordered melody corpus, this population is saved. A new random melody corpus is generated to start the experiments. The graphical tool used to create the shapes shown in Fig. 2 has a limit of length 42 (between opposing diagonal corners), hence the target integer vectors were filled with random integers between 0 and 42. Each random target was used in a GE run to create an initial corpus of 30 melodies to start the experiments.

In each full cycle a melody is replaced; after 30 cycles the entire corpus has been replaced with melodies created by the system. Each cycle involves a full evolutionary run to create an Agent. For each of these runs the population was of size 100 (seeded by the 100 Agents evolved in the initialisation phase) run over 100 generations. A plot of the best and average fitnesses achieved across 100 generations is shown in Fig. 4. This plot shows a rapid decrease in best fitness which tapers off around generation 16 and becomes more stable after generation 50. The average fitness remains higher which indicates the population is still diverse at the end of a run. Only a single best Agent is chosen as a fitness function for the following Melody run.

The average of the best fitnesses across 30 Melody runs is also shown in Fig. 4. While on average the best fitness does taper off over generations it is clear that there are large differences in the values obtained (note the $\log_{10}$ scale) resulting in large peaks particularly towards the beginning of a run. These differences are a result of the syntax of the Agent (fitness function). For example Agent:

```
-1D1+3D2*4D3-0D4+2D5-2D6+5D7*4D8*4D9*1D10+4D11+5D12-5D13-3D14-1D15.
```

contains several multiplication terms which will lead to very large results for a number of melodies. As the system always evolves with minimising fitness, these melodies will be eliminated over time but these wide ranges are an inherent part of the system. Such large differences made it infeasible to include the average fitnesses on the same plot.

4.3 Melodies

This paper presents a study based around algorithmic composition, although composing 'good' melodies or compositions better than other, more focussed systems was not the main aim of the study. This study is based on the concept of creating a self-adaptive cyclical system creating subjective fitness functions that make decisions in a justified, reproducible and explainable manner. Nevertheless, it is an implemented compositional system and as such we offer the reader a number of short melodies available at http://ncra.ucd.ie/Site/loughranr/evo_2017.html. Hex1, Hex2, Alt1, Alt2, Return1 and Return2 are examples of melodies created from the shapes shown in Fig. 2 used to create the initial Agent population. Each melody is played twice to help the ear recognise the given shape. The Hex melodies

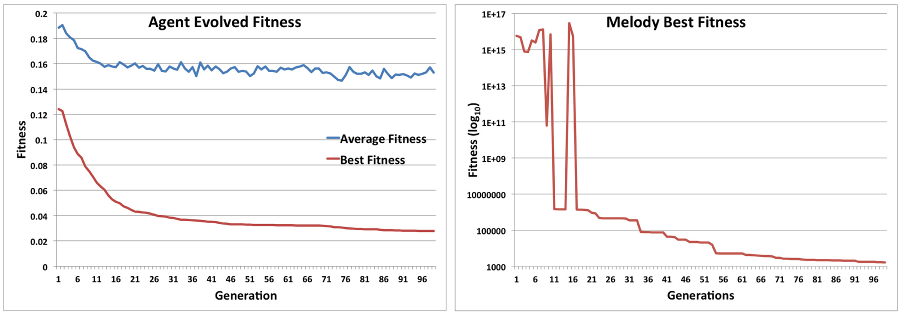

Fig. 4. Average vs. best fitness of the evolution of the evolving Agents and the best fitness of the evolving Melodies averaged over 30 runs.

display a series of slightly altering segments that return to the original segment such as in an hexagonal (or ideally circular) shape, the Alt melodies alternate between two distinct styles of segments and the Return melodies start with a segment, change it somewhat and return to the original a number of times. For completeness we have included two sample Random melodies.

It is worth noting that the average fitness obtained when creating the preparatory corpus of melodies was 5.69 whereas the average fitness obtained when creating the Random melodies was 22. This is as to be expected as, although the randomly created distances were defined to be within the ranges of those of the shapes, the relative distances between them were not controlled to be graphically meaningful and so may not have been possible to achieve. It was observed by listening to the melodies from the initial Random corpus that some of these melodies compensated for this by simply repeating individual segments. Thus there is a wide variety within the Random melodies with some containing elements of repetition such as Random1, whereas others such as Random2 contain no such discernible patterns. This is acceptable in the proposed experiment, as all that is required to start the experiment is a varied population of melodies represented in segments. Whether or not they now conform to specific shapes is of no consequence once the full evolutionary run is started.

Four short Evolved melodies are included as a demonstration of the output of the final system. Each is again played twice in succession. From listening to the final outputs, it is clear that some of the final melodies have kept remnants of the original shapes whereas others have not. Evol45 for example clearly displays an Alternating pattern. Evol0 on the other hand (the first melody created by the system) does not display any such pattern. Thus the best Melodies, and subsequent best Agents, have learned from a corpus that was originally full of well patterned melodies, yet such knowledge is sometimes apparent in the output and sometimes not.

Melody20 is a longer composition created from the final (melody) population of the last cycle. For this composition the top 20 melodies were selected according to the fitness given by the current Agent. A distance metric was calculated between each of these melodies by calculating the Levenshtein distance between the pitch contours (at each time-step) of each pair of melodies. These 20 melodies

were then grouped according to melodic similarity and concatenated together in this grouping as one extended melody. This was in an attempt to create a smooth transition between individuals; individuals that are melodically similar would be grouped together creating a smooth transition between musical ideas rather than an abrupt jump between individuals in a diverse population. The top 20 individuals are clustered in this case as:
[[0, 1, 2, 4, 7, 9], [3, 10], [5, 8], [16, 17, 12, 15, 18], [11], [14], [20], [6, 13], [19]]
Unsurprisingly, four out of the top five 'top' melodies are clustered together — at the end of an evolutionary run, the population has converged and it is likely that the top few individuals are identical or very similar. After that, however, it is evident that the melodies are grouped more in relation to similarity than in fitness. There is, however, some similar content audible in a number of individuals. Much of the content in the top individual can be heard to re-surface many times throughout this composition. This is to be expected in an evolutionary run and we have previously considered it to be of benefit in using evolutionary methods for composition as this 'similar yet different' aspect of parts of a melody can lead to variation on a theme, which is known to be a pleasant quality in music [14]. In future studies we plan to investigate more interesting ways of traversing through the population for the creation of well-formed compositions.

4.4 Discussion

An important concept throughout an experiment such as this is to consider how data or knowledge is being transferred through the system. The initial Agent population is created using a clearly patterned database (melodies evolved to emanate a certain shape) but such patterns are not directly input into the system again at any point. The population was created in such as way as to ensure the individuals had some meaning — that each Agent was not completely naive but was created to start with some innate ability. Thus the system is not provided with an explicit target for the fitness function; the initial population is created from a preparatory run using a known target and the ability learned from this preparatory run is maintained within the initial population and propagated through the system in an indirect manner. The system adapts to its own response in creating new Agents and subsequent new melodies once the full cycle is started. Hence the system is self-sustainable and runs without any external human input, once it has been initialised.

In each cycle, the population was initialised using the same Agents created in the preparatory run, with all other parameters remaining the same; the only difference between cycles is one different melody in the corpus. The stochastic elements of EC methods combined with this one change resulted in a different Agent population and hence new melody in each cycle. This demonstrates that even in controlled experiments with very few degrees of freedom, evolutionary methods still have the power to develop numerous varied results that satisfy the proposed criteria. This search ability and flexibility is one of the reasons EC methods are so suitable for the development of creative systems.

Furthermore, the transformation of knowledge or ability throughout an experiment such as this, is reminiscent of Boden's third suggestion of how computers may be creative as discussed in the Introduction: the 'transformation' of ideas. The proposed system transforms ideas and information many times throughout its operation: in the transformation from genome-phenome through the use of the grammar, in the passing of clustering ability through from the preparatory step to the evolution of the clustering Agents and in the cyclical employment of the evolved Agent in evolving new melodies. Knowledge transformation is a key concept throughout this proposed system for it is through the transformation and abstraction of data, knowledge and ideas that true creativity can emerge.

The Lovelace Test for creativity states that for a system $\boldsymbol{A}$ with output $\boldsymbol{o}$ and human architect $\boldsymbol{H}$, the system can only be deemed to be creative if $\mathbf{H}$ cannot explain how $\mathbf{A}$ created $\mathbf{o}$ [5]. While on the surface this may seem an easy test to pass, on closer inspection it is remarkably difficult — if at all possible. By default the architect (or programmer) — assuming they understand their own code — will be able to explain how the system created the resultant output. The system proposed in this study is certainly still explainable, but it was created in a way that the knowledge gained and behaviour displayed by the system was abstracted an extra level away from the human architect. We hope that further studies into conceptual, knowledge transferring and self-organising systems may assist in the development of computational creative systems or creative AI.

Although this system does produce melodies (discussed in Sect. 4.3), at this stage of development we have not conducted any human-evaluations on these melodies. The system at the moment represents a theoretical computational study; the focus of this study is on the description and proposal of the ideas within the system rather than an adjudication of the output. Furthermore, we acknowledge that at the moment, the melodies produced are short and not particularly impressive. We believe this can be improved in the next phase of the system that will run in a similar cyclical and adaptive manner, but could be employed with an improved grammar that can create more sophisticated melodies. As a compositional system we are interested in gauging human response to the results. Future work on more sophisticated version of the system will involve a comprehensive survey-based set of human evaluations.

5 Conclusions

This paper presents a cyclical algorithmic compositional system based on mutually dependent runs of GE. The system creates a best Agent, evolved using a corpus of melodies, which is subsequently used as a fitness function to create a new melody that replaces a melody in the corpus and the process is repeated. This best individual Agent is evolved according to the way in which it clusters the melodies, and how much its clustering correlates with the average clustering of the population of Agents. Each initial population of Agents is populated with a set of Agents trained in a preparatory step on a corpus of melodies whose ideal clustering is pre-defined. In this way, the Agents take some learned ability and transfer it into the system. No other outside influence is introduced to the system once it has started. A number of melodies created during various stages of the system were presented.

Although an implemented system is presented accompanied by a number of produced melodies, we still consider this system to be of more theoretical than practical interest at this stage. The strength of the proposed system lies in the data transformation and knowledge transfer that is described throughout the paper. Immediate future work will consider further analysis of the running of the system over a number of cycles to determine if there is any predictable behaviour exhibited by the system and what this may imply. We plan to develop the system to include a more sophisticated grammar that would enable more interesting musical compositions to be created. The key challenge in this development will be to improve the compositional ability of such a system while explicitly maintaining the autonomy of the entire system. We feel that the strength and novelty of this study lie in the autonomy of a self-adaptive complex system applied to a subjective task and, as such, it is imperative that this autonomy is maintained and preserved in future implementations.

Acknowledgments. This work is part of the App'Ed (Applications of Evolutionary Design) project funded by Science Foundation Ireland under grant 13/IA/1850. We would like to thank the anonymous reviewers for their suggestions on how to improve the paper.

References

1. Biles, J.A.: Straight-ahead jazz with GenJam: a quick demonstration. In: MUME 2013 Workshop (2013)
2. Boden, M.A.: The Creative Mind: Myths and Mechanisms. Psychology Press, New York (2004)
3. Boden, M.A.: Computer models of creativity. AI Mag. **30**(3), 23 (2009)
4. Brabazon, A., O'Neill, M., McGarraghy, S.: Grammatical evolution. In: Brabazon, A., O'Neill, M., McGarraghy, S. (eds.) Natural Computing Algorithms, pp. 357–373. Springer, Heidelberg (2015)
5. Bringsjord, S., Bello, P., Ferrucci, D.: Creativity, the turing test, and the (better) lovelace test. In: Moor, J.H. (ed.) The Turing Test, pp. 215–239. Springer, Heidelberg (2003)
6. Cook, M., Colton, S.: Generating code for expressing simple preferences: moving on from hardcoding and randomness. In: Proceedings of the Sixth International Conference on Computational Creativity June, p. 8 (2015)
7. Dahlstedt, P.: Autonomous evolution of complete piano pieces and performances. In: Proceedings of Music AL Workshop. Citeseer (2007)
8. Eigenfeldt, A., Pasquier, P.: Populations of populations: composing with multiple evolutionary algorithms. In: Machado, P., Romero, J., Carballal, A. (eds.) EvoMUSART 2012. LNCS, vol. 7247, pp. 72–83. Springer, Heidelberg (2012). doi:10.1007/978-3-642-29142-5_7
9. Göksu, H., Pigg, P., Dixit, V.: Music composition using genetic algorithms (GA) and multilayer perceptrons (MLP). In: Wang, L., Chen, K., Ong, Y.S. (eds.) ICNC 2005. LNCS, vol. 3612, pp. 1242–1250. Springer, Heidelberg (2005). doi:10.1007/11539902_158

10. Hickinbotham, S., Stepney, S.: Augmenting live coding with evolved patterns. In: Johnson, C., Ciesielski, V., Correia, J., Machado, P. (eds.) EvoMUSART 2016. LNCS, vol. 9596, pp. 31–46. Springer, Cham (2016). doi:10.1007/978-3-319-31008-4_3
11. Jenks, G.F.: The data model concept in statistical mapping. In: International Yearbook of Cartography, vol. 7(1), pp. 186–190 (1967)
12. Lehman, J., Stanley, K.O.: Efficiently evolving programs through the search for novelty. In: Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation, pp. 837–844. ACM (2010)
13. Loughran, R., McDermott, J., O'Neill, M.: Grammatical evolution with zipf's law based fitness for melodic composition. In: Sound and Music Computing Conference, Maynooth (2015)
14. Loughran, R., McDermott, J., O'Neill, M.: Tonality driven piano compositions with grammatical evolution. In: 2015 IEEE Congress on Evolutionary Computation (CEC), pp. 2168–2175. IEEE (2015)
15. Loughran, R., McDermott, J., O'Neill, M.: Grammatical music composition with dissimilarity driven hill climbing. In: Johnson, C., Ciesielski, V., Correia, J., Machado, P. (eds.) EvoMUSART 2016. LNCS, vol. 9596, pp. 110–125. Springer, Cham (2016). doi:10.1007/978-3-319-31008-4_8
16. Loughran, R., O'Neill, M.: Generative music evaluation: why do we limit to 'human'?. In: Computer Simulation of Musical Creativity (CSMC), Huddersfield, UK (2016)
17. Miranda, E.R.: On the evolution of music in a society of self-taught digital creatures. Digit. Creativity **14**(1), 29–42 (2003)
18. Munoz, E., Cadenas, J., Ong, Y.S., Acampora, G.: Memetic music composition. IEEE Trans. Evol. Comput. **20**(1), 1–15 (2016)
19. Pearce, M., Wiggins, G.: Towards a framework for the evaluation of machine compositions. In: Proceedings of the AISB 2001 Symposium on Artificial Intelligence and Creativity in the Arts and Sciences, pp. 22–32. Citeseer (2001)
20. Scirea, M., Togelius, J., Eklund, P., Risi, S.: MetaCompose: a compositional evolutionary music composer. In: Johnson, C., Ciesielski, V., Correia, J., Machado, P. (eds.) EvoMUSART 2016. LNCS, vol. 9596, pp. 202–217. Springer, Cham (2016). doi:10.1007/978-3-319-31008-4_14
21. Shao, J., McDermott, J., O'Neill, M., Brabazon, A.: Jive: a generative, interactive, virtual, evolutionary music system. In: Chio, C., et al. (eds.) EvoApplications 2010. LNCS, vol. 6025, pp. 341–350. Springer, Heidelberg (2010). doi:10.1007/978-3-642-12242-2_35
22. Stanley, K.O., Lehman, J.: Why Greatness Cannot Be Planned: The Myth of the Objective. Springer, Heidelberg (2015)
23. Thywissen, K.: GeNotator: an environment for exploring the application of evolutionary techniques in computer-assisted composition. Organised Sound **4**(02), 127–133 (1999)
24. Todd, P.M., Werner, G.M.: Frankensteinian methods for evolutionary music. In: Griffith, N., Todd, P.M. (eds.) Musical Networks: Parallel Distributed Perception and Performace, p. 313. MIT Press, Cambridge (1999)
25. Waschka II, R.: Composing with genetic algorithms: GenDash. In: Miranda, E.R., Biles, J.A. (eds.) Evolutionary Computer Music, pp. 117–136. Springer, Heidelberg (2007)