Evolving Musical Sequences with N-Gram Based Trainable Fitness Functions

ManYat Lo, Simon M. Lucas, University of Essex, UK

Abstract—Conventionally, automatic music composition is done by evolving music sequences whose fitness is evaluated by a human listener. This interactive approach has led to interesting results but is very time consuming. Here we propose a system that is capable of automatically generating music using an evolutionary algorithm (EA), replacing the human evaluation process with a trainable music evaluation algorithm. This algorithm can be trained on existing music samples, such as Mozart compositions for example. This kind of system could provide a fast and cheap music composition tool. The current evaluation system is implemented with an N-gram language model. This paper discusses the system in two parts. Firstly, it describes the performance of the proposed music evaluation algorithm. Secondly, it discusses the impacts of different sequence-oriented genetic operators in the evolutionary algorithm. Part one of the experimental results show that the N-Gram model is able to distinguish the composer of piano compositions by Mozart, Beethoven and Chopin with up to 81.9% accuracy. Part two of the results show that some of the sequence-oriented operators increased the fitness of the generated melodies, but some operators did not. The impacts of these operators are discussed in the experimental results section. Significantly, the results also show that better classification accuracy does not necessarily lead to better evolved music, suggesting that perceptual relevance is also an important factor.

I. Introduction

ALGORITHMIC music composition is an important topic in music technology. It could be described as a set of well-defined rules of music composition. The idea of algorithmic composition is not new. Researchers Hiller and Isaacson [8] were possibly the first to study this field scientifically. They based their research on Markov Chains. Since then many researchers have attempted to address different problems of algorithmic composition. The later developments of algorithmic composition can be divided into stochastic and deterministic. Mozart's dice game [2] is an example that used a stochastic technique. This game uses a dice roll as a random generator to select prewritten bar music and these were combined to form full length music compositions. All prewritten bars were composed such that every bar harmonises with every other one, hence, the resulting music satisfies certain musical requirements. Deterministic systems often pre-define all the rules, with the output then depending only on the particular inputs used to drive the generator (as opposed to some stochastic element). The Fibonacci sequence [4] and L-systems [16] are examples of a deterministic technique.

The main objectives of this research are to investigate the capability of N-grams to act as the music fitness function in an evolutionary algorithm, and to investigate the effectiveness of different sequence-oriented variation operators for this task.

II. Background Knowledge and Literature Review

A. N-grams

In the field of Natural Language Processing, N-grams are a widely used statistical language modelling technique, a type of Markov Model (MM). This technique is based on the assumption that the next symbol in a string is largely dependent on a small number of immediately preceding symbols in the string. N is a model parameter which represents the total number of preceding symbols plus the current symbol. N could be any number >= 1. E.g. N=1 is called a Uni-gram which looks zero (N-1) symbols into the past. N=2 is called a Bi-gram which looks only one symbol into the past. N=3 is called Tri-gram which looks two symbols into the past. Despite (alternatively, because of) their simple nature, N-grams work well in practice, providing sufficiently large samples of training data are available. N-grams have been mostly used in text prediction, text retrieval and pattern recognition [9]. Recently, researcher Doraisamy [5] used N-gram for music indexing in a large music database which contains polyphonic music in MIDI format. Statistical methods tend to be more robust in learning from data samples than purely symbolic techniques. Generally working with N-grams, user chooses N preceding symbols that the next symbol is assumed to depend on. Equation (1) is the equation for a Bi-gram model (N=2); this generalises in straightforward ways to different values of N.

w_n : Individual symbol w_n, where n is it order.

w_{n-1} : Previous symbol to w_n.

$w_1, w_2, w_3 \ldots, w_n$ Or w_1^n : Representing the complete string (sequence) of symbols.

$P(w_1^n)$: The probability of complete string.

$$P(w_1^n) = P(w_1)\prod_{k=2}^{n} P(w_k \mid w_{k-1}) \qquad (1)$$

One important detail of applying N-grams in practice is the estimation of the probability of events that did not occur in the training data. Naïve estimation techniques based solely on a frequency count would give these zero probability, meaning that any test sequence that contained such an event would also be given zero probability. To prevent this, we use the simple 'add one' discounting method, where all events are given an initial occurrence count of 1. Refer to [11] for more N-gram details.

B. Maximum Likelihood Sequences

An interesting feature of Markov models is that it is possible to compute the maximum likelihood sequences (MLS). That is, having chosen the length of the sequence, a dynamic programming algorithm can be used to find the sequence of symbols that is most likely, given the model. Hence, when using a trained N-gram model as a fitness function in an evolutionary algorithm, we can efficiently compute the *fittest* possible sequence of a given length. This allows us to compare how close the EA gets to this optimum. As we shall see, the best sounding sequences tend to have a mid range fitness somewhere between the low fitness of purely random sequences and the high fitness of the MLS, which tends to be very repetitive.

C. Algorithmic Composition

The idea of Algorithmic Composition with Artificial Intelligence (AI) techniques is relevant to Music Information Retrieval (MIR) systems [17], [20], and involves classification or pattern-learning tasks. An MIR system searches and compares the musical similarity between the query and the target pieces in a large music database (usually a collection of Music Instrument Digital Interface (MIDI) files). The major challenge in music classification applications is the automatic classification of musical styles and ranking the similarity between musical pieces. There are two main difficulties encountered. Firstly, melody extraction: how should the music be represented as input to the system. The main issues to consider here are how to deal with polyphony and note duration. Secondly, MIDI files contain multiple tracks which may represent multiple instruments in the same composition. In this paper we define "melody" to mean a sequence of notes. Some multiple-track music consists of more than one instrument playing simultaneously or with percussion, but only some of the notes are recognised by the listener. It is necessary that only the recognisable notes from the polyphonic music are selected for melodic evaluation.

The major challenge of algorithm composition is defining an evaluation function (i.e. the fitness function). Four categories of fitness function are listed below.

1. Interactive fitness function: Traditional GA/GP fitness function is algorithmic, but this interactive approach uses human to evaluate each chromosome. Biles [1] developed an interactive jazz jamming system. The system generated melodies by chord and scale mappings techniques, and used a GA with musically meaningful genetic operators to evolve solo melodies through the interactive fitness function. Unehara & Onisawa [21] implemented a similar system which generated 16-bar melodies. The generation of initial population and genetic operators were based on music theory. Two main drawbacks of the interactive fitness function approach are subjectivity and inefficiency. Due to subjectivity, an individual user biases his / her decision corresponding to previous listening or changes his mind over time because of his emotional state (this can also be seen as an advantage depending on one's perspective). Regarding inefficiency, users must listen to all the potential music (solutions) in order to evaluate the whole population, which severely limits the number of fitness evaluations that can be made. On the plus side, this approach helps musically unskilled users compose their own music and reflects the user's musical taste.

2. Human-trained fitness function: Johanson [10] developed an interactive GP system with automatic fitness raters. This system generated short musical sequences which were then evaluated by a set of fitness raters. Their automatic fitness raters were based on neural networks with shared weights trained with the back propagation algorithm. A user rates a list of melodies while the raters use the result to learn to rate a melody in a similar way to the user. Tokui [19] developed a drum pattern generative system called "Conga". Conga was implemented with both GA and GP techniques. GA individuals represent short pieces of drum patterns, and GP individuals represent the arrangement of these short patterns. Both populations were evolved interactively through the user's evaluation. In a later version of his system, he used neural network to model the human musical subjective behaviour and then applied it into the GA's fitness function. Even though this approach increases the evaluation process speed, the subjectivity still remains a black-box. Another problem identified by Todd & Werner [18] is that if the musical search space has steep surfaces (thus, if changing one note can make a melody go from good to bad), a neural network may fail to model this adequately.

3. Knowledge-based fitness function: This approach embeds domain-specific knowledge into the fitness function. McIntyre [13], Bach in a box: The evolution of four–parts baroque harmony used genetic algorithm; Grachten [7] developed a Jazz Improvisation Generator (JIG), which generated jazz melodies by combining musical constraints with the probability distribution of notes. Three major constraints were used: tonality, the improvisation must be tonal to the key of the music; continuity, the melodic contour of the improvisation must be smooth, not convoluted;

structure, interrelated groups of notes are identified and used. Finally, that system is evaluated by the same three constraints. Papadopoulos & Wiggins [14] developed a jazz melody generative system with a GA. Eight different musical characteristics were embedded into the fitness function to evaluate each chromosome. Three musically meaningful genetic operators and two standard crossover genetic operators were implemented. Phon-Amnuaisuk and other researchers [15] developed a four-part homophonic system which was based on rich knowledge. Domain-specific knowledge was encoded in the GA's chromosome representation, reproduction operators and fitness function. All generated harmonisation melodies were marked by a music lecturer, according to the criteria which he used for 1st year undergraduate students' harmony test. Most of the output earned a mark around 30%. This low mark was due to the lack of coherent large-scale musical progression. But they claimed their system was felt to be better than student harmonisers at getting the basic rules right. Knowledge-based approaches offer efficiency, with sound structure and knowledge built into the system which means more structured musical output, but the trade-off is less novelty. Also the system is largely dependent on the domain-specific knowledge, which must be explicitly implemented correctly and clearly in the system. However, such knowledge might be subjective to the designer. This is a major drawback for knowledge-based systems.

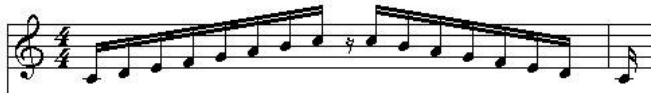
4. Fitness functions learned from data: This approach employs probabilistic methods to capture musical patterns or a set of rules in a score. Composers usually define a set of musical grammars to compose music pieces. N-gram models and Hidden Markov Models (HMM) are the most popular techniques for evaluating music pieces. Typically these models evaluate the likelihood of a musical sequence given the trained model. Downie & Nelson [6] developed and evaluated a music information retrieval system which implemented the N-gram indexing technique. The system dealt with musical queries by the user, whereby the parameters entered by the user are traced and compared for the exact or similar music in a large folk music database. Each folk song is converted into an interval-only representation of monophonic melody. The interval-only representation ignored individual note duration. Doraisamy [5] used N-grams for the development of a polyphonic music retrieval system for large music collections. The main objective of the Doraisamy investigation was to use the N-gram approach for searching fully polyphonic music data. McCormack [12] implemented a music composition system based on the L-Systems [16] approach. Pitch, note duration and timbre are encoded to the grammar symbols. The system generated a sequence of notes by a set of probabilistic re-writing rules. But all the re-writing rules were pre-defined by the user. Chai & Vercoe [3] developed an automatic music classification system that classifies folk music to its respective country of origin. The system was based on an HMM. The aim was to investigate the significant statistical differences among folk music from different countries and to compare the classification performances using different melody representations.

III. IMPLEMENTATION

Our Automatic music composition (AMC) system uses an evolutionary algorithm to evolve melodies which are represented as sequences of integers for evolution, and then converted to standard MIDI format for playback. Typically, 250 notes is set as default length of each melody. The system consists of two parts, the N-gram style classifier and the EA-based music evolution system. The user gives a list of music samples to train the N-gram which becomes a fitness function embed in the EA. The trained N-grams have discriminative power to assess melodies, and give fitness value to each melody. Then the EA generates a pool of random melodies and evolves them.

The extracted and evolved melodies are monophonic, not polyphonic. It is much more difficult to extract musical information from polyphonic music. It involves multiple pitches such as chords playing simultaneously, though in practice the onset of each note can be delayed. Also, percussion is not compatible with the representation of a tonal instrument.

In our system, music melodies are represented as sequences of integers. We used two different encoding methods. 1) Each integer value represents a single absolute pitch, except for 128, which represent the rest note. The integer value of pitch follows the standard MIDI format. 2) Each integer value represents an interval between notes (interval can be + or -). In musical term +1 interval is equal to raise one semi-tone. Rest note is not a pitch, so we used a special symbol +128 to represent it. We set the pitch different from rest note to next pitch is = next note pitch – previous pitch note. All notes are of the same duration, but the rest note can be used create notes of different perceived duration. Figure 3a is an example of music shown in integer string format with both encoding methods.



(1.Absolute pitch) Integer String = [60, 62, 64, 65, 67, 69, 71, 72, 128, 72, 71, 69, 67, 65, 64, 62, 60].

(2. Pitch different) Integer String = [+2, +2 , +1, +2, +2, +2, +1, +128, +0, -1, -2, -2, -2, -1, -2, -2].

Figure 3a – C Major Scale

Listing 3a shows the details of these operators. Hence, we constructed four experiments. Ten operators were defined as below:

Listing 3a - Sequence Oriented Variation Operators.

0) Random single note modification: randomly select a location in a sequence and randomly change the note value.
1) Random segment note modification: similar to operator 0), but modifies each note value in the selected segment.
2) Random segment copy and paste: randomly selects two segments, copies the first segment and replaces it into second segment.
3) Random segment reversion: randomly selects a segment, sort into reverses note ordering.
4) Random segment transpose: randomly selects a segment, and then randomly transposes each note value, the transpose range from -5 to +5.
5) Random segment swap: randomly selects two segments, and then swaps them.
6) Random segment ascending: randomly selects a segment, sort into ascends note ordering.
7) Random segment descending: randomly selects a segment, sort into descends note ordering.
8) Random segment copy from training samples: randomly selects a segment from training sample and then copies and replaces it into a segment from the sequence which is selected randomly.
9) Observed distribution single note modification: randomly select a location of sequence and modifies the note by sampling from observed note distribution in the training data.

Note: a segment ranges from 2 – 10 notes, based on our intuition of what would be reasonable.

IV. A. Experiments, Results and Analysis

Four experiments were constructed: 1. N-gram classifier discrimination experiment. 2. Maximum Likelihood Sequence (MLS) experiment. 3. Music specific genetic variation operators experiment. 4) Pitch different Bi-gram model experiment. Note that experiment 1, 2, and 3 used absolute pitch N-grams models. Experiments 1 and 2 provide insight into the properties of the N-Gram models used, while experiments 3 and 4 test the ability of an EA to optimise sequences with respect to the N-Gram fitness functions, and also investigate how musical the evolved sequences sound. For the EA we used a random mutation hill-climber (RMHC), with the wide selection of variation shown in Listing 3a. We also experimented with population-based EAs, but these did not significantly outperform the RMHC.

A. Experiment 1: Composer Classification

This is an experiment that analyses the N-gram model's discriminative power by a standard pattern classification evaluation technique – leave one out cross validation. The motivation behind this experiment was the idea that a classifier with the discriminative power to classify the compositions of three composers of the same genre might also be good at as a fitness function for music evolution. The N-gram classifier consists of three individual N-grams, and each one is assigned a particular composer, Mozart, Beethoven and Chopin. For personal music style classification tasks, the composers are assumed to have equal prior probabilities, and each test melody is assigned to composer whose N-gram model gave rated the sequence as having the highest likelihood.

In the experiment, 282 classical music samples were used for training the N-grams. There are 110 Mozart, 72 Beethoven and 100 Chopin (Approximately 470,000 notes). Table 4a is the experiment results summary of a Bi-gram composer style classifier which applied the add-one smoothing technique. In short the add-one smoothing is a method to smooth out the Bi-gram probability matrix. It avoids the large impact of unseen data, and performs well over sparse data, as the total of training songs is 282 only. For more details refer to [11]. The result shows that the average correct rate = 231/282 = 81.9%. In addition we calculated the confidence intervals (assuming that each classification event can be treated as the flip of a biased coin). Hence, we can say the actual average correct rate lies within the interval of 77.8% - 85.3% with 90% confidence. The Uni-gram results are shown in table 4b. It achieved an average correct rate = 218/282 = 77.3%, the confidence interval is 72.9% -81.1%.

Table 4a
Confusion Matrix: Bi-gram with add-one smoothing

		Predicted		
		Mozart	Beethoven	Chopin
Actual	Mozart	93	16	1
	Beethoven	10	60	2
	Chopin	0	22	78

Table 4b
Confusion Matrix: Uni-gram with add-one smoothing

		Predicted		
		Mozart	Beethoven	Chopin
Actual	Mozart	92	17	1
	Beethoven	17	50	5
	Chopin	1	23	76

We were surprised to see that the Uni-gram performed nearly as well as the Bi-gram. When the methods fail, they fail in different ways. Bi-grams fail because they over-fit the data, while Uni-grams fails because they under-fit it.

B. Experiment 2: Maximum Likelihood Sequence

The maximum likelihood sequence (MLS) was computed

for two reasons: because it gives a value for the best possible fitness given the trained model, and in order to listen to this sequence. The Viterbi dynamic programming algorithm was used to generate the MLS [11]. A simple random mutation hill climber (RMHC) was used to evolve a single melody with Mozart Bi-gram fitness function. The RMHC was run for 600,000 generations (iterations), length of chromosome was 250 and for each mutation, operator 0 was used (randomly select and modify a single note in the sequence). Using this method it fails to get close to the MLS. Then we compared MLS fitness and the evolved melody fitness. The experiment result is shown in Figure 4a.

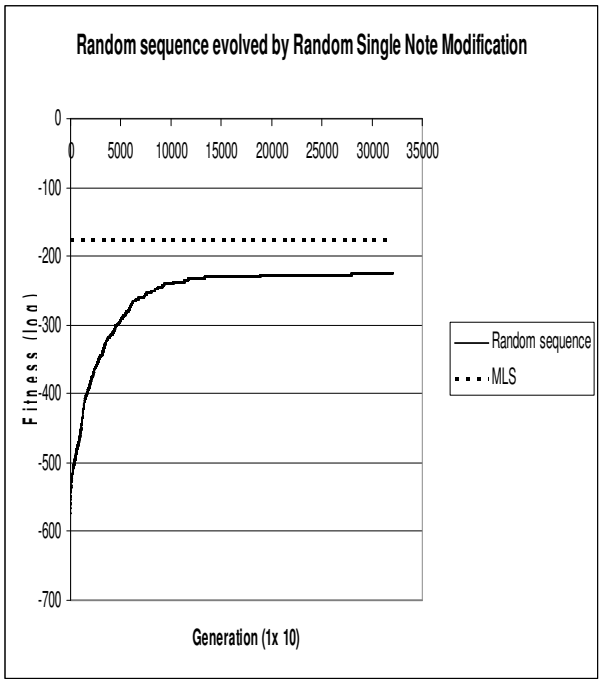


Figure 4a: The maximum likelihood sequence fitness = Log -177.26, melody = 77 79 81 79 81 79 81 79…… (Repeat)…77 76 74 128. The initial random melody fitness = Log -576.19, after 600000 generations it reduced to Log -225.98. Note: the fitness after 350000 remained the same.

The pattern of maximum likelihood sequence is based on two repetitive notes. This happens with the Bi-gram model because there must be one or more transitions with the highest fitness value (e.g. [81][79]= log -0.5918), therefore a sequence containing these two state transitions with repeating order would be the most probable sequence that the system could generate. The reason that the last four notes in the MLS was not repeated because the fitness of sequence (77 76 74 128 77) = Log -3.1286 is lower than the fitness of sequence (79 81 79 81 79) = Log -2.8683.But when both sequences had the last note removed the fitness of sequence (77 76 74 128) = Log -1.8814 is higher than the fitness of sequence (79 81 79 81) =Log -2.2764, therefore the (77 76 74 128) would be the optimal last four notes. The Maximum Likelihood Sequence technique finds the circular state chain that has the highest fitness over other circular state chains. Then it just keeps repeating that highest fitness state chain to construct a maximum likelihood sequence.

In figure 4a, the sequence started to converge after generation 100000. The optimal sequence fitness is lower than MLS, it was trapped in a local optimum. Its fitness differ from MLS fitness Log -48.72 (Log-225.98 – Log -177.26), about 27.4% of MLS fitness. The reason that the optimal sequence did not converge to the MLS is the evolutionary system is heavily dependant on the genetic operator, and with operator-0 the search space contains many local optima. As an example, if the MLS is (79 81 79 81) and the initial sequence is (80 64 20 50) in the first generation, the operator might modify the sequence to (80 64 79 50), which is closer to the MLS in hamming distance, but has a lower fitness than the initial sequence.

When listening to the evolved sequence, it sounds more interesting than the MLS. A reasonable variety of notes produces interesting sound. But too much variety of notes sounds random and irritating, or even boring. The sound of MLS is boring, as it keeps repeating two notes only.

C. Experiment 3: Sequence-oriented variation operators

Performance of the evolutionary system is heavily dependant on its genetic operators. These operators are responsible for modifying each chromosome at each generation. For a musical evolutionary system, it is important to evaluate each musically meaningful operator. The aim of this experiment was to analyse the performance of each operator given the Bi-gram and Uni-gram fitness functions.

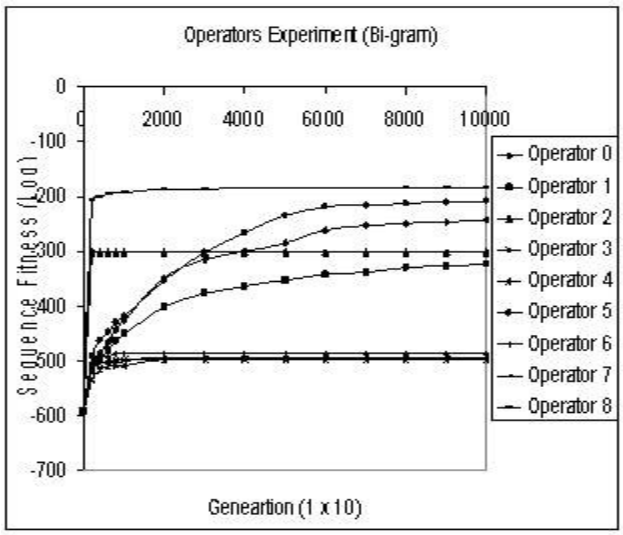


Figure 4b: Performance of various operators given the Bi-gram fitness function.

Experiment 3 has the same setup as experiment 2, except that 100000 generations were used because of the harder

fitness function. Figure 4b and 4c show the experiment al results. The summary of results is shown in table 4c and the performance ranking summary is shown in table 4d.

Note: Nine operators are individually run.

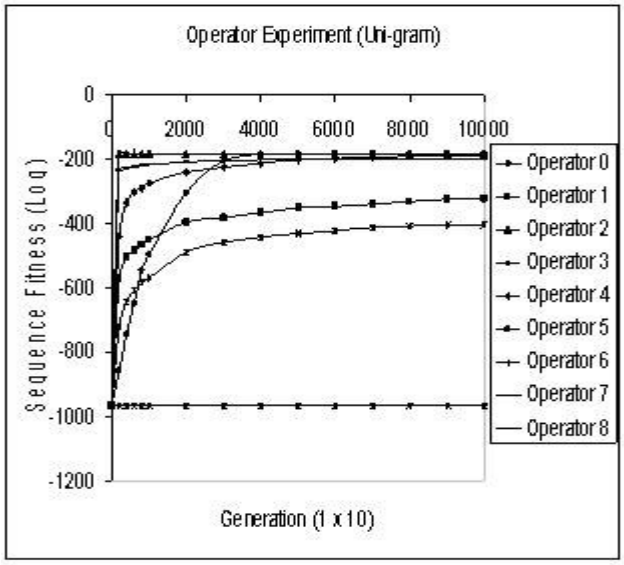


Figure 4c. Performance of various operators given the Uni-gram fitness function.

Table 4c

O.N = Operator number. O.S.F = Optimal sequence fitness. Note: the MLS with Bi-gram is Log -177.26 (same to experiment 2), the MLS with Uni-gram is Log -182.18. The initial sequence fitness with Bi-gram is Log -590.57. The initial sequence fitness with Uni-gram is Log -965.75.

O.N	O.S.F in Bi-gram	O.S.F in Uni-gram
0	Log -243.12	Log -187.51
1	Log -322.91	Log -322.99
2	Log -302.03	Log -182.18
3	Log -498.08	Log -965.75
4	Log -493.75	Log -404.89
5	Log -209.22	Log -182.18
6	Log -486.76	Log -965.75
7	Log -493.93	Log -965.70
8	Log -183.40	Log -199.56

Table 4d

Ranking	Bi-gram	Uni-gram
1	Operator 8	Operator 5
2	Operator 5	Operator 2
3	Operator 0	Operator 0
4	Operator 2	Operator 8
5	Operator 1	Operator 1
6	Operator 6	Operator 4
7	Operator 4	Operator 3
8	Operator 7	Operator 6
9	Operator 3	Operator 7

The actual sequence of MLS with Uni-gram is: sequence = 128 128 128… (Repeat)….128. Integer value 128 represents a Rest note in the system. The frequency of rest note in the samples is the highest. The following analysis describes the convergent power of each operator when applied to the Bi-gram and Uni-gram fitness functions.

Operator 0 – This operator performs well on both Bi-gram and Uni-gram. The Uni-gram model is well suited to this operator, as the evolved sequence is close to the MLS. The fitness of the optimal sequence is Log -187.51, very close to the MLS fitness Log -182.18. This is because Uni-gram model takes individual state into account only, no relation between previous or next state. Using this operator on Uni-gram model is will converge to the MLS given enough generations. The Uni-gram fitness landscape is convex and similar to the well studied one-max problem. Using this operator with the Bi-gram fitness function gives much poorer performance, and the fitness landscape has many local optima.

Operator 1 – This operator modifies a group of notes, the convergent power is lower than the operator 0 which modifies a single note value at each generation. That is because each time when the operator randomly modifies a group of notes, it is very hard to pick up the higher fitness notes or notes transition. If the fitness of the modified sequence is less than the fitness of unmodified sequence, then the modification is discarded, the sequence remain unchanged

Operator 2 – The convergent power of this operator is very high when applied on Uni-gram, low when applied on Bi-gram. This is because for Bi-gram, the note ordering is very sensitive. Hence, it is very hard to randomly copy a group of notes that has a higher fitness value when these notes fit into the target location of the sequence. However, Uni-gram does not have this problem, as note ordering does not affect the fitness of sequence on Uni-gram.

Operator 3 – This operator is the worst performing on Bi-gram experiments because this operator does not change any single value of note. It reverses the note ordering, does not modifies the note ordering in a different way, it slows down the evolving sequence to converge to the MSL. This operator is even worst when it performs on Uni-gram model, as it takes no effect of the sequence fitness. That is because the only way to change the sequence fitness on Uni-gram is to modify the note value. Note ordering is no affect on Uni-gram model.

Operator 4 – The transpose operator did not perform well on both models, this is little surprise that because this operator modifies each note value in the segment. The reason of this is because the sequence allows overwriting only when the new modified sequence has a higher fitness; if it is lower, the sequence remains same. For example the operator only allows +5 and -5 transpose, if a chosen segment never get the higher fitness within +5 and -5 transpose, thus this segment would never change the note value at all generation.

Operator 5 – The swap operator swaps two segments locations, in additional to that it does overwrite the segment. For example if the locations of the first segment and second

segment are the same, then the first segment is overwritten on top of the second segment. This operator rearranges the note ordering on Bi-gram model displaying high convergent power. For Uni-gram model, this operator copies the existing high fitness note and overwrites the lower fitness note, thus the convergent power is high too.

Operator 6 and 7 – These sorting operators did not perform well but acceptable on Bi-gram. They sort the sequence in one way note ordering only. When a song contains ascending or descending note ordering in the instrument solo section and then it sounds interesting. But these operators do not affect anything on the Uni-gram model, since these operators do not change the value of any note in the sequence.

Operator 8 – This operator performs well on both models that are because this operator copied a segment from the training samples. Therefore any segment that is copied is seen by the classifier beforehand and fitness of each copied segment is guaranteed to have high fitness.

After analysing both experiment 3(a) and 3(b), we can draw some conclusions: 1) The convergent power of an operator is heavily dependant on the different models (Bi-gram or Uni-gram). 2) The operator that randomly changes the note value would cause the convergence to slow down, but it adds more variety to the note value. 3) The operator that copies the high fitness of note value or ordering would cause the convergence to quicken but results in less variety of notes.

D. Experiment 4 – Interval Statistics

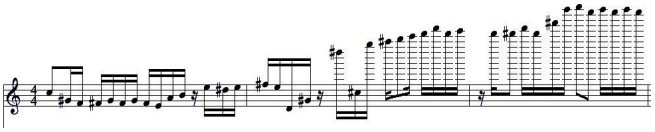
The pitch difference Bi-gram model experiment consisted of two parts. Part 1 comprised the classification tests, conducted in the same way as experiment 1 but using a pitch difference Bi-gram model. It achieved an average correct rate = 199/282 = 70.57%, the 90% confidence interval is 65.91% - 74.83%. Part 2, this experiment used the pitch difference Mozart Bi-gram to generated ten melodies, and evolved them by observed distribution of single note modification genetic operator (operator 9). Those ten generated melodies had fitness values are in range from Log -290 to Log -250. After the evolution process, the evolved melodies fitness values are in range from Log -170 to Log -150. The summary is shown in table 4e.

Table 4e

Melody	Fitness before evolution	Fitness after evolution
Melody 1	-285.41	-166.79
Melody 2	-282.93	-158.09
Melody 3	-281.81	-162.21
Melody 4	-277.16	-163.05
Melody 5	-284.65	-157.93
Melody 6	-254.74	-155.20
Melody 7	-271.24	-161.43
Melody 8	-279.08	-157.54
Melody 9	-273.99	-160.00
Melody 10	-273.77	-165.90

The average correct classification rate of the pitch difference Bi-gram model is lower than the absolute pitch Bi-gram model. But there are two advantages when using this model. 1. It can be used to discriminate melodies which are in different musical key/scale. 2. This model is general enough to generate a variety of notes with high fitness value. This is because it does not capture specific notes but instead the interval between them. For example, suppose there are two melodies (C-major and D-major scales) containing notes: C D E F G A B and D E F# G A B C# D. Their fitness values are completely different under absolute pitch N-gram model, but are exactly same under pitch difference N-gram model. All the major scales are constructed by the same interval arrangement (+2,+2,+1,+2,+2,+2,+1). Figure 4d shows some generated and evolved melody segments.

Segment 1: Evolved by GA with absolute pitch Bi-gram model (operator 8), Bar 1-3.



Segment 2: (Melody 5) Generated by pitch difference Bi-gram model Bar 1-3.

Segment 3: (Melody 5 after evolution) Evolved by GA with pitch difference Bi-gram model (Operator 9), Bar 1-3.

Figure 4d

Segment 1 is the melody evolved using the absolute pitch Bi-gram model. We described the reason for this repetitive sequence earlier. Most of the notes are repeated without much variety. Segment 2 contains a variety of notes and exhibits more (perhaps excessive) variation in pitch. In segment 3, the sequence of notes is constant and smooth, no high interval skipping. The sequence of melody 5 after evolution contained variety of notes as much as melody 5 before evolution, but the notes arrangement is constant, as up and down stairway. Note that the overall melody fitness value is higher after the evolution process. In general, the pitch difference model generates melodies with a variety of notes and keeps high fitness values. In this case the actual melody which evolves using the pitch difference model is more interesting than the melody evolved using the absolute pitch model.

V. Conclusion and Further Direction

Our work provides a fundamental investigation of using N-grams as trainable music evaluators (fitness functions) in an evolutionary algorithm. Experiment 1 and Experiment 2 (part 1) have demonstrated that N-gram classifiers were able to correctly identify the composer around 80% of the time.

Our most optimistic hope was that this would enable the evolution of new pleasant sounding melodies, similar to those composed by some of the great composers. An interesting result here is that while both random note sequences are uninteresting, and the maximum likelihood sequence (MLS) is also uninteresting, evolutionary algorithms can be used to explore the space between those extremes and find melodies that are reasonably interesting to listen to.

For experiment 3 we analysed each musical operator in detail, showing their performance on both Bi-gram and Uni-gram models. There was no single operator that could evolve a sequence that reaches the MLS in both models. Most of the evolved sequences are in the range of fitness value Log -200 to Log -500. As mentioned before, given the N-gram model, the creation of interesting melodies relies on the *inability* of the algorithms to converge to the optimum.

In experiment 4 results show that using the pitch difference Bi-gram model for melody generation and evolution subjectively outperforms the absolute pitch representation, even though they are no better at classification. We suggest the reason for this is that using interval (successive note pitch difference) statistics is closer to a perceptual model than using absolute pitch statistics. This is intuitively obvious, since melodies played in a different key sound essentially the same, and have identical likelihoods in the pitch difference model, but completely different ones in the absolute pitch model.

We propose that an interesting way forward from here is to continue using pitch difference N-grams to capture the statistics of a musical style, but then evolve parameters for a musical generation system that incorporates some general musical knowledge. In this way, it may be possible to get closer to the goal of evolving interesting music based largely on a trainable fitness function. It would also be interesting to experiment with longer range or variable length N-grams.

Finally, we are conducting some listening tests in order to get some idea of how the evolved music sounds to human listeners (see http://www45.brinkster.com/amcsystem/SongRating.asp) and we encourage readers to visit the site and rate the sequences.

REFERENCES

[1] J. A. Biles, "GenJam: A genetic algorithm for generating jazz solos," In Proceedings of the 1994 International Computer Music Conference, 1994.

[2] J. A. Biles, "Composing with sequences:…but is it art?," Information Technology Department, Rochester Institute of Technology.

[3] W. Chai, & B. Vercoe, "Folk music classification using hidden markov models," In Proceeding International Conference on Artificial Intelligence, 2001.

[4] E. D. Dobson, Understanding Fibonacci Numbers. Traders Press, 1984.

[5] S. Doraisamy, "Polyphonic music retrieval: The n-gram approach," PhD thesis, 2004.

[6] S. Downie, and M. Nelson, "Evaluation of a simple and effective music information retrieval method," Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2000.

[7] M. Grachten, "JIG: Jazz improvisation generator," In Proceedings of the MOSART Workshop on Current Research Directions in Computer Music, 2001.

[8] L. Hiller & L. Isaacson, Experimental Music. New York: McGraw Hill, 1959.

[9] F. Jelinek and R.L. Mercer, "Interpolated estimation of markov source parameters from sparse data," Proceedings of the Workshop on Pattern Recognition in Practice, pp. 381-97, North Holland, Amsterdam, 1980.

[10] B. Johanson and R. Poli, "GP-Music: An interactive genetic programming system for music generation with automated fitness raters," Genetic Programming: Proceedings of the Third Annual Conference, 1998

[11] D. Jurafskyl and J. H. Martin, Speech and Language Processing, An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition. International Edition. Prentice Hall, 2000.

[12] J. McCormack, "Grammar based music composition," In Complex Systems 96: From Local Interactions to Global Phenomena, R Stocker et. Al., eds, ISO Press, Amsterdam, 1996.

[13] R. A. McIntyre, "Bach in a Box: The evolution of four part baroque harmony using the genetic algorithm," International Conference on Evolutionary Computation, 1994, pp. 852-857.

[14] G. Papadopoulos and G. Wiggins, "A genetic algorithm for the generation of jazz melodies," Proceedings of STeP 98, Jyväskylä, Finland, 1998.

[15] S. Phon-Amnuaisuk, A. Tuson and G. Wiggins, "Evolving musical harmonisation," International Conference on Artificial Neural Networks and Genetic Algorithms, Slovenia, 1999.

[16] P. Prusinkiewicz and A. Lindenmayer, The Algorithmic Beauty of Plants. New York: Springer, 1990.

[17] P. Salosaari & K. Jarvelin, "MUSIR – A retrieval model for music," University of Tampere, Department of Information Studies, Research Note RN-1998-1, 1998.

[18] P. M. Todd and G. M. Werner, "Frankensteinian approaches to evolutionary music composition," In N. Griffith and P.M. Todd (Eds.). MIT Press, 1998.

[19] N. Tokui and H. Iba, "Music composition with interactive evolutionary computation," 3rd International Conference on Generative Art, Milan, 2000.

[20] A. Uitdenbogerd and J. Zobel, "Melodic matching techniques for large music databases," Proceedings of the seventh ACM international conference on Multimedia (part 1), 1999, pp.57-p66.

[21] M. Unehara and T. Onisawa, "Construction of music composition system with interactive genetic algorithm," Proceedings of 6th Asian Design International Conference, Tsukuba, Japan, 2003.