Automating the Composition of Popular Music: The Search For a Hit

Timothy Millea supervised by Jonathan Wakefield

University of Huddersfield, HD1 3DH, England, UK
{t.a.millea, j.p.wakefield} @ hud.ac.uk

Abstract. Automated composition may be regarded as a search within the space defined by a datatype representing musical compositions. We develop a hierarchical representation of popular music compositions with the aim of increasing the probability of finding potential hits. Musical variations are calculated as difference vectors between patterns extracted from a given set of existing compositions. These form the basis of the mutation operator within an evolutionary algorithm search.

Key words: automated composition, demotic, evolutionary algorithm, lead sheet, popular music, representation, search space.

1 Introduction

"Music is the universal language of mankind"

(Henry Wadsworth Longfellow)

The term *popular music* is often associated with music recorded and bulk-distributed for profit. However music has been popular, in the sense of appealing to, or being liked by many people since long before the days of bulk-distribution. Evidence suggests that every known culture has made and listened to music since the emergence of modern humans around two hundred thousand years ago [1]. The term therefore does not define a particular genre or style of music. Rather, popular music may be considered *demotic*, meaning *of the people*: it is accessible to the people and subject to ever evolving popular tastes.

Perhaps it is because classical composition has been formalised in music treatises, e.g. Fux's 1725 treatise on counterpoint is an early example[2], that much of the research in the area of automated music composition has focussed on classical forms. Computers may be programmed to generate new music by satisfying the set of constraints that a music treatise encodes [3] or by interpreting the treatise as a set of production rules [4] by which new musical sentences are formed [5][6][7]. However, the inherent difficulty of these approaches is that of formulating or deriving constraints or rules that are sufficient to admit music that is desired and yet exclude music that is not.

Recent work has investigated automatically deriving a set of rewrite rules to hierarchically represent a given piece of music. By attaching probabilities where

a rule yields more than one possible rewrite, such a system may be used to generate new music [8].

Attempts have been made to formalise popular music, e.g. [9] states that the most successful popular music structure is *verse-chorus-verse-chorus-bridge-chorus*. However, different popular music genres have different common musical features. A treatise on popular music may fail to capture some common features in a given genre while inadvertently hard-coding inappropriate restrictions. Even within one genre, the constant evolution of popular musical tastes makes it likely that today's treatise will not capture the musical fashions of tomorrow.

We propose an approach that does not rely upon any formalism of the musical genre concerned. Instead, musical features are derived from existing compositions given as input and new music is generated with similar features. The interfaces of the system's input and output are electronic scores in Musical Instrument Digital Interface (MIDI) file format. Audio processing, lyrics, performance and production are therefore outside the scope of the current work. This paper represents the first year of progress of a three year investigation.

2 Approach: The Search for a Hit

A datatype defines, by its dimensions and range of all possible values, a space that contains those values. A datatype to represent musical compositions therefore defines a space that contains not just all existing musical compositions but all potential future compositions too. Theoretically, this space need only be searched to find new, as yet unwritten music.

We wish to design a datatype to represent popular music compositions. we first consider that the ideal representation corresponds to a space that contains only potential hit songs. The search in this case would never fail. The worst-case search space is infinitely large and contains no music. Between these extremes there are many possible representations corresponding to varying densities of good solutions. One such example is standard music notation which is capable of representing any tonal musical composition in addition to an infinite number of patterns of notes without any recognisable musical structure or accepted musical meaning. The greater the density of potential hit songs in the search space, the easier the task is of finding them. The approach is therefore to develop a representation that corresponds to the smallest search space achievable that is still capable of adequately representing popular music compositions.

An evolutionary algorithm will be employed to search the space defined by the representation. An evolutionary algorithm search is *robust* [10] and has attractive analogies with the processes of human creativity. The evolutionary algorithm operator of combination (crossover) has its dual in creative *influence* as either the confluence of two or more ideas or the abstraction of an idea and application outside its original context. The mutation operator corresponds to *chance discovery*. The iterations of an evolutionary algorithm as it converges towards a solution represent *refinement* towards a goal. The fitness function encapsulates how the output is judged, independently of how it was achieved.

2.1 Evolutionary Algorithm

Figure 1 shows one possible conceptual evolutionary algorithm for an automated popular music composer. The given set of existing compositions are converted to an internal representation. These are combined in pairs and the resulting offspring mutated and assessed for fitness. The fittest songs form a new population and the process repeats until a sufficiently fit song is found. This is converted to MIDI and the algorithm terminates. The work will investigate other possible evolutionary searches.

```
form the initial population from the given set of input songs
REPEAT
   FOR the number of offspring songs DO
      pick two songs
      COMBINE them to form a new song
      MUTATE the new song
      evaluate the FITNESS of the new song
      SELECT a new population of songs
UNTIL a sufficiently fit song is found
```

Fig. 1. Conceptual evolutionary algorithm for the automated composer

2.2 Research Challenges

Key to the outlined approach is developing a suitable representation of popular music compositions. It must be sufficiently general-purpose to admit a wide variety of popular music genres but ideally be free of *ambiguity*, i.e. any given composition has only one possible representation, and of *redundancy*, i.e. not admit non-musical values. These ideals form the guiding principles of the development of the representation, irrespective of whether they are wholly achievable.

The evolutionary algorithm operators, *combine* and *mutate*, must be developed in tandem with the representation such that the result of their application has the expected musical meaning, e.g. the combination of two songs should produce a new, complete song that inherits the musical features of both its parents.

Suitable objective and heuristic measures will need to be developed to form the fitness function.

The input and output of the system are electronic scores in MIDI format. There is therefore the requirement to be able to convert between MIDI and the internal representation.

3 Lead Sheet Minimalism

The search space may be reduced by eliminating from the representation anything not essential to a composition. Here the design choices are informed by

considering that there are many possible different ways of performing the same composition e.g. with different instruments, in different keys and tempos, even relative timings and pitches may be subtly altered but the composition remains recognisably the same.

The essential parts of a popular music song are often conveyed to session musicians by means of a *lead sheet* which specifies only the lyric, melody and chords. Musicians rely upon their knowledge of the musical genre and style to interpret a lead sheet but still the lead sheet representation is sufficient to convey the essence of the composition, whether it is composed by human or by computer. Lyrics are outside the scope of this project. This leaves melody and harmony as a reasonable minimum essence of a popular musical composition. Key may be removed from the domain of popular music composition as it is common practice to change the key to suit the range of the lead vocalist.

Melody may be minimally represented as a sequence of notes expressed in terms of pitch intervals relative to the tonic and duration as relative, quantised time units. Harmony may be minimally expressed as a sequence of harmonic functions, equivalent to chords relative to the key. Table 1 shows the seven harmonic functions rooted by the seven pitch classes of a given major scale, together with the corresponding chords in the key of C major. This choice of seven harmonic functions is considered a reasonable compromise between the four main functions used by Yogev for automatic harmonisation [11] and the relatively unrestricted eighty-four employed by GenJam for automated jazz solo improvisation [12].

Table 1. Harmonic functions in a major key

Function	Name	e.g. in C maj key
I	Tonic	C maj
ii	Parallel subdominant	D min
iii	Parallel dominant	E min
IV	Subdominant	F maj
V	Dominant	G maj
vi	Parallel tonic	A min
vii	Dominant 7th (incomplete)	G7

A given popular musical composition may thus be reduced to a sequence of triples: *(pitch interval, quantised relative duration, harmonic function)*.

4 First Order Representation

The basic unit of repetition in music is the bar. In many genres of popular music the bar captures the basic rhythmic pulse. All the bars in any one song are generally of equal duration relative to tempo. Here, the choice of absolute tempo and variations of it may conveniently be regarded as aspects of performance and

not of composition. In the absence of explicit tempo or meter changes, a song is therefore composed of equal-sized bars which are subdivided into a constant number of equal-sized time units.

4.1 Time Unit Box System

An alternative notation to traditional music score notation often used to express rhythmic patterns is the Time Unit Box System (TUBS) developed by Philip Harland in 1962 [13]. It is now also commonly used in step sequencers, drum machines and the like. TUBS subdivides a repeated sequence into equal duration units which are marked as having a rhythmic beat or not. Using the TUBS notation, it is easy to represent a rhythmic variation as a change of state of one or more boxes. Figure 2 shows a TUBS representation of the two-bar 'Bo Diddley' rhythm and a variation upon it.

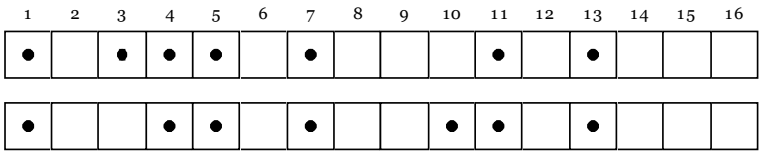

Fig. 2. The Time Unit Box System (TUBS) representation of the Bo-Diddley rhythm and a variation beneath

4.2 Rhythmic Distance

This work adapts the Kolmogorov variational distance metric using the Temporal Elements Displayed as Squares (TEDAS) system as described by Toussaint [13]. The distance between two rhythmic patterns is calculated graphically. The horizontal axis is time units. Beat onset times are marked on the horizontal axis and a square is inserted between each pair. The two patterns of squares are overlaid and the variational distance is the difference in their areas. Toussaint reports that this method of calculating rhythmic distance agreed the best with human perception from those he surveyed [13]. Figure 3 illustrates the calculation of the distance between the two rhythms from figure 2 with the shaded areas representing the difference.

4.3 Melodic and Harmonic Distance

Ó Maidín proposed a method near-identical to the one above to measure *melodic distance* [14] between two equal-length phrases. The vertical component, instead of being equal to the duration of the current note, is pitch class, i.e. A-G# without octave information. The horizontal axis is time units as above. Melodic distance is the sum of the differences of the resulting areas.

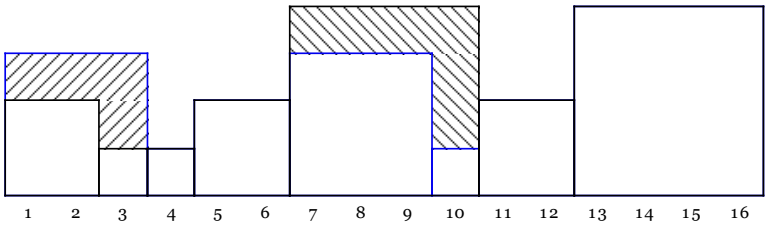

Fig. 3. Calculating the Kolmogorv variational distance between two rhythmic patterns

Here we adopt Ó Maidín's melodic distance metric and adapt it to measure harmonic distance, i.e. the distance between two sequences of harmonic functions. The choice of one harmonic function per scale degree (section 3) enables a relatively simple calculation of the distance between two same-length chord sequences. The vertical component of harmonic distance may be calculated from the interval between the roots of the harmonic functions with reference to the harmonic series, i.e. in ascending order: unison, octave, perfect fifth, perfect fourth, major third, minor third and so on in declining pitch intervals. For example, the difference between tonic and dominant harmonic functions (C major and G major chords in the key of C major) is a perfect fifth yielding a value of two, as the second harmonic.

4.4 Music Roll Datatype

The above methods of calculating rhythmic, melodic and harmonic distances inform the design of the representation. The lead-sheet triple *(pitch interval, quantised relative duration, harmonic function)*, and TUBS-based representations are combined. A musical composition is divided into a finite number of quantised time-units that subdivide the regular rhythmic pulse of the music. Instead of just beat onset as in TUBS, each time unit has three components corresponding to rhythm, pitch, and harmony. The presence in a time unit of any of these three components corresponds to note onset, change of pitch or change of harmonic function respectively. The melodic contour may thus be described independently of rhythm. An empty time-unit represents a continuation of the previous state. It is the datatype equivalent of the music roll in that time units form one dimension and a set of pitches form the other. In the interests of simplicity, there is no explicit 'note-off' or silence value. This music roll datatype forms the first order representation.

5 Mutation as Musical Variation

Using the music roll notation, the difference in two patterns may be represented as a vector offset from one to the other. The vector represents the variation one pattern is of the other. Vectors may then be applied to patterns other than those

from which they were derived in order to achieve new and musically meaningful variations. The set of difference vectors derived from the input musical patterns form the basis of the evolutionary algorithm mutation operator.

The severity of a mutation is ideally controlled in relation to the overall success over a number of mutations, where a successful mutation is defined as one that results in increased fitness. The underlying principle is that in a random population, far away from the optimal solution, there is an expectation that half of all mutations are successful. However, as the optimal is approached, there are fewer opportunities for improvement, so the average success rate falls until the mutation severity exceeds twice the fitness distance from the optimal and all mutations are unsuccessful. Figure 4 illustrates this phenomenon.

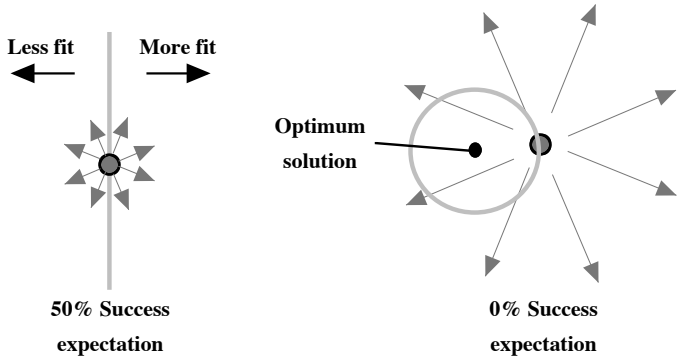

Fig. 4. How relative mutation severity affects mutation success

Mutations are therefore differences between patterns found in the input music. Their severity is measured using the adapted Kolmogorov and appropriate mutations applied to new offspring songs in order to control the convergence of the search.

6 Repetition

Popular music typically contains repetition at different levels of scale, e.g. rhythms and repeated chorusses. A piece of music may therefore be represented more concisely as a higher level sequence of repeated patterns. Consider the following sequence.

```
abaaabacabaaabaaabacabaa
```

It contains repetitions of the two subsequences *abaa* and *abac*. Let $A = abaa$ and $B = abac$ then the original sequence may be represented by the higher order pattern $ABAABA$. This itself contains the pattern ABA twice. Let $C =$

ABA then the original sequence reduces to CC. Clearly in music, this type of hierarchical representation corresponds to a top-down description of musical structure [15]. What is the algorithm to *decompose* a given musical sequence into such a hierarchy ? The required process is much like grammatical generation [4] in reverse, where each occurrence of a subsequence is replaced with an identifier, cf. a non-terminal symbol.

The statistical distributions of patterns in a sequence may be calculated by building a variable degree Markov model similar to those employed in the Continuator [16] of maximum degree equal to half the length of the total sequence. No repeating pattern can be longer than half of the total. The Markov model gives the frequency of occurrence of every subsequence found in the original upto this maximum length. The *repetitive significance* of each pattern may then be calculated from its length and frequency.

Consider a sequence of n notes in which the same subsequence of p notes occurs f times. The new symbol covers $f \times p$ notes. However, the new symbol must occur f times in place of the original pattern and require a reference to the original pattern which is of length p, yielding a cost of $f + p$. We define *repetitive significance* as coverage per unit cost, as given in equation 1.

$$RS = \frac{f \times p}{f + p} \tag{1}$$

This equation is highly related to the one based on information theory given in [8] where the objective is maximise the information per symbol.

Referring to the example sequence, the pattern with the highest repetitive significance may now be identified.

```
input sequence = abaaabacabaaabaaabacabaa
(f x p) / (f + p) maximised by abaa
```

After the most repetitive pattern has been identified and substituted, the second most repetitive pattern is identified and substituted and so on until the entire sequence is covered and replaced by a new sequence of symbols. This is the second order representation. It is a sequence of symbols each of which corresponds to a pattern of notes found in the original input sequence.

```
ABAABA where
   A = abaa, B = abac
```

The same algorithm may be recursively applied, i.e. to any find patterns of patterns at increasing orders, until there is no further repetition.

```
D where
   D = CC where
      C = ABA where
         A = abaa, B = abac
```

The result is top-down, hierarchical description of the original input sequence capturing repetition on every scale found in the original input sequence. At the lowest level are repeated sections of music roll, i.e. patterns of notes and harmonic functions, while the higher levels represent their hierarchical organisation.

This algorithm has been successfully implemented and initially applied to the percussion parts of several Stevie Wonder MIDI song files. In all cases the algorithm was able to identify the basic rhythmic patterns of a small integer multiple number of bars in length, and the major sections of each song. The initial implementation of the algorithm often ranked different rotations of the same rhythmic pattern equally, i.e. it was unable to align the start of the pattern with the start of a bar. An example of equal-scoring rhythm patterns found in the song Superstition is shown in table 2. An analysis of the results suggests that an irregular first or last bar were sufficient to cause this behaviour. It may readily be corrected for by biasing for a beat at the start of the pattern.

Table 2. Equal-scoring rhythm patterns from Stevie Wonder's Superstition

Score	Pattern	Time intervals
2415	/.....	[24]
2415	./.....	[1,23]
2415	../.....	[2,22]
2415	.../.....	[3,21]
2415	..../.....	[4,20]
2415	...../.....	[5,19]
2415	...../.....	[6,18]
2415	...../.....	[7,17]

7 Combination

The purpose of combination or crossover in an evolutionary algorithm is to mix elements of two or more trial solutions in the expectation that some offspring are fitter than their parents. Here we wish to combine the features of two songs in a musically meaningful way.

There are a number of possible means of combining the data structures outlined above. These are similar to hierarchical crossover techniques [17]. Firstly, as in genetic algorithms, sequences from each parent may be simply spliced together at a randomly chosen point in the corresponding order. In order to retain musicality, this should be at a *point of similarity* [17], e.g. the same harmonic function. Secondly, some of the non-terminal symbols in one order of one song may be redirected to refer to patterns in the corresponding order below in the other song. A simple example at the highest level would be the replacement of the chorus of song A by the chorus from song B. At the lowest level it could be the replacement of a repeated bar or line in one song by one from the other.

This type of combination will be termed *cross-indirection* and may be partial as per these examples or total where the offspring randomly acquires the entire sequence from the corresponding level of one of the parents. Finally, elements or entire sequences may jump levels, e.g. a second order pattern in one parent may be promoted to a third order pattern in the offspring. In each case, where the corresponding number of symbols differs between the parents there will an added conforming step to ensure no reference is made to a non-existing pattern.

8 Conclusion and Future Work

We have presented an approach to the generation of new music based on an evolutionary search of a space defined by a representation of popular music compositions. Consideration has been given to reduce the size of the search space that the representation defines in order to increase the density of good solutions. Its hierarchical properties capture the patterns of repetition on different scales. Musical variations are extracted as difference vectors between patterns in the input music to form musically-meaningful mutations. Distance metrics for rhythm, melody and harmony variations control the convergence of the evolutionary algorithm towards its fitness goals. Work is currently underway to refine the mutation and combination operators.

Future work will investigate what combination of similarity measures between existing and trial compositions and high level goals such as memorability, complexity and sing-ability will best serve as a measure of compositional fitness.

References

1. Wallin N.L., Brown S., Merker B., The Origins of Music, ISBN 0262731436 (1963)
2. Fux J.J., Mann A. (translator): J.P.: The Study of Counterpoint. W. W. Norton & Company ISBN 0393002772 (1965)
3. Pachet F., Roy P.: Musical Harmonization With Constraints: A Survey. Constraints 6:7-19 (2001)
4. Chomsky N.: Three Models of Language. IRE Transactions in Information Theory 2:113-124 (1956)
5. Holtzman S.R.: A Generative Grammar Definition Language for Music. Journal of New Music Research 9:1-48 (1980)
6. Roads C., Wieneke P.: Grammars as Representations for Music. Computer Music Journal 48-55 (1979)
7. Steedman M.J.: A Generative Grammar for Jazz Chord Sequences. Music Perception 2:p52 (1984)
8. Thornton C.: Hierarchical Markov Modelling for Generative Music. International Computer Music Conference , Montreal, Canada (2009)
9. Blume J.: 6 Steps to Songwriting Success. Billboard Books ISBN 0823084221 (1999)
10. Goldberg D.E.: Genetic Algorithms in Search, Optimisation and Machine Learning. Addison-Wesley (1989)
11. Yogev N., Lerch A.: A System for Automatic Audio Harmonization. 25th Tonmeistertagung VDT International Convention, Leipzig, Germany (2008)

12. Biles, J.: GenJam: A Genetic Algorithm for Generating Jazz Solos. Proceedings of the International Computer Music Conference, ICMA, San Francisco (1994)
13. Toussaint G.: A Comparison of Rhythmic Similarity Measures. Proc. International Conference on Music Information Retrieval 242245 (2004)
14. Ó Maidín D.: A Geometrical Algorithm for Melodic Difference. Computing in Musicology, 11:6572 (1998)
15. Marsden A.: Automatic Derivation of Musical Structure: A Tool for Research on Schenkerian Analysis. Proc. International Conference on Music Information Retrieval (2007)
16. Pachet F.: The Continuator: Musical Interaction With Style. New Music Research Journal 32 p 333 (2003)
17. Bentley P.J., Wakefield J.P.: Hierarchical Crossover in Genetic Algorithms. Proc. 1st On-line Workshop on Soft Computing (1996)