Towards melodic extension using genetic algorithms

Michael Towsey
Queensland University of Technology
128 PUBLICATIONS 1,272 CITATIONS
SEE PROFILE

Andrew R. Brown
Griffith University
119 PUBLICATIONS 837 CITATIONS
SEE PROFILE

Susan Wright
University of Melbourne
98 PUBLICATIONS 503 CITATIONS
SEE PROFILE

Joachim Diederich
Psychology Network Pty Ltd
188 PUBLICATIONS 2,423 CITATIONS
SEE PROFILE

Educational Technology & Society 4(2) 2001
ISSN 1436-4522

Towards Melodic Extension Using Genetic Algorithms

Michael Towsey, Andrew Brown, Susan Wright and Joachim Diederich
Queensland University of Technology
Kelvin Grove, QLD 4059, Australia
m.towsey@qut.edu.au
a.brown@qut.edu.au
s.wright@qut.edu.au
j.diederich@qut.edu.au
Tel: +617 3864 3285
Fax: +617 3864 1963

Abstract

Genetic Algorithms (GA's) are considered promising for music composition because they combine ‘creativity’ (ability to explore a large search space) with constraints (creative 'excess' is 'pruned' using a fitness function). A major difficulty with the use of GA's for this task is to define fitness functions which capture the aesthetic qualities of the wide range of successful melodies. In this paper we report on research that addresses this problem in the context of a modest compositional task, melodic extension. We describe 21 melodic features used as the basis for a GA fitness function and for mutation procedures. We discuss how the features were chosen, measured for significance, and might be incorporated into a fitness function.

Keywords

Genetic algorithms, Music composition, Creativity, Fitness functions, Melodic extensions, Melodic features

Introduction

The use of heuristic principles for automated music composition is well established in computer music (Moorer, 1972, Laske, 1992) but the use of Genetic Algorithms in this task is less well established. GA's have been used previously in the harmonisation of melodies [McIntyre, 1994; Horner and Ayers, 1995; Wiggens et al., 1997] and for autonomous composition as distinct from computer-assisted composition [Horner and Goldberg, 1991; Jacob, 1995]. An interactive system was developed by Biles, [1994] for performance of jazz improvisations.

Our research extends the role of GA's into computer-assisted composition. Our goal is to assist secondary music students in learning the art of melodic composition by working with a computer system that actively suggests extensions to, and provides critiques of, the student’s composed melody. Our study explores the suitability of Genetic Algorithms to the modest, yet still difficult task of generating appropriate extensions to melodic 'seeds' composed by students. These melodic suggestions will provide ‘timely’ interactive support to the students. Success in the project depends in large measure on the construction of a suitable fitness function.

The CUE software, developed by Cope [1997], provides a similar function to that aimed for in our project - it generates extensions to partially composed material in common music notation. Our project differs from Cope’s work in its use of Genetic Algorithms rather than Augmented Transition Networks (ATN) for melodic generation. We believe that GA's can provide a system at least as flexible as CUE, with stable performance, and can avoid the problems of scalability inherent to using databases with ATNs. However, unlike Cope, we limit our task to melodic extension, that is, we make no attempt to harmonise.

Systems for the algorithmic composition of music can be conveniently categorised into three types; rule based systems, systems which learn by example and genetic algorithms [Todd & Werner, 1999]. Rule based systems vary widely in the complexity of their rules. Very simple systems have a history going back even to Mozart, with attempts being made to disguise the simplicity of the underlying algorithm. More recently, Ebcioglu [1984] has developed an expert system for harmonisation of chorals, which incorporates the complex rules of counterpoint and voice-leading. Rule based systems are not necessarily deterministic. For example, function generators used to produce sound sequences can have randomly initialised parameters.

The advantage of rule based systems is that their encoded knowledge is explicit. They behave predictably which, of course, contributes to their utility. However systems that attempt to emulate human performance based on rules tend to be brittle, that is, there is usually some set of circumstances in which the system breaks down. The difficulty is that, to the extent human performance can be defined by rules, there are exceptions to the rules and exceptions to the exceptions. Such knowledge is very difficult to encode explicitly. For example, a complete and consistent explication of the rules of counterpoint and voice leading has yet to be achieved [Loy, 1991].

Systems which learn by example typically include Markov models [Loy, 1991] and recurrent neural networks (for example, the CONCERT software [Mozer et al., 1994]). When trained on melodic sequences, these systems learn statistical knowledge about local structure - that is given a sequence of four or five notes the system is able to predict the probability of any potential succeeding note. For a Markov model, this information is encoded in state transitions and for a neural networks in its weights. The advantage of such systems is that they can produce tuneful sequences, at least on a short time scale. Since they do not learn large scale structure, their melodies give no sense of direction - they meander.

The EMI and CUE software [Cope, 1997], attempts to overcome the limitation of a single Markov model by implementing a hierarchy of such models known as an Augmented Transition Network. Network models high in the hierarchy learn large scale structure, while those low in the hierarchy learn local structure. Such a system nevertheless requires considerable manual crafting.

The difference between rule-based systems and learning systems is best highlighted by an example. If we wish to construct a system for choral harmonisation that avoids parallel fifths, then a rule-based system would explicitly contain a rule to avoid parallel fifths and would check for their occurrence. To ensure that a neural network never produced a parallel fifth, however, we would simply ensure that an instance of one never occurred in its training examples.

The first published record of the use of GA's for music composition is Horner and Goldberg [1991]. Progress over the subsequent decade is reviewed by Burton and Vladirova [1999] and Todd and Werner [1999]. The significance of the GA approach to music composition is best appreciated by reflecting on an important conflict that all AI systems must resolve when simulating human performance, the trade-off between *novelty* and *structure*. In short, we want the system to follow the structure of human performance, but we want it to surprise us with novelty when appropriate!

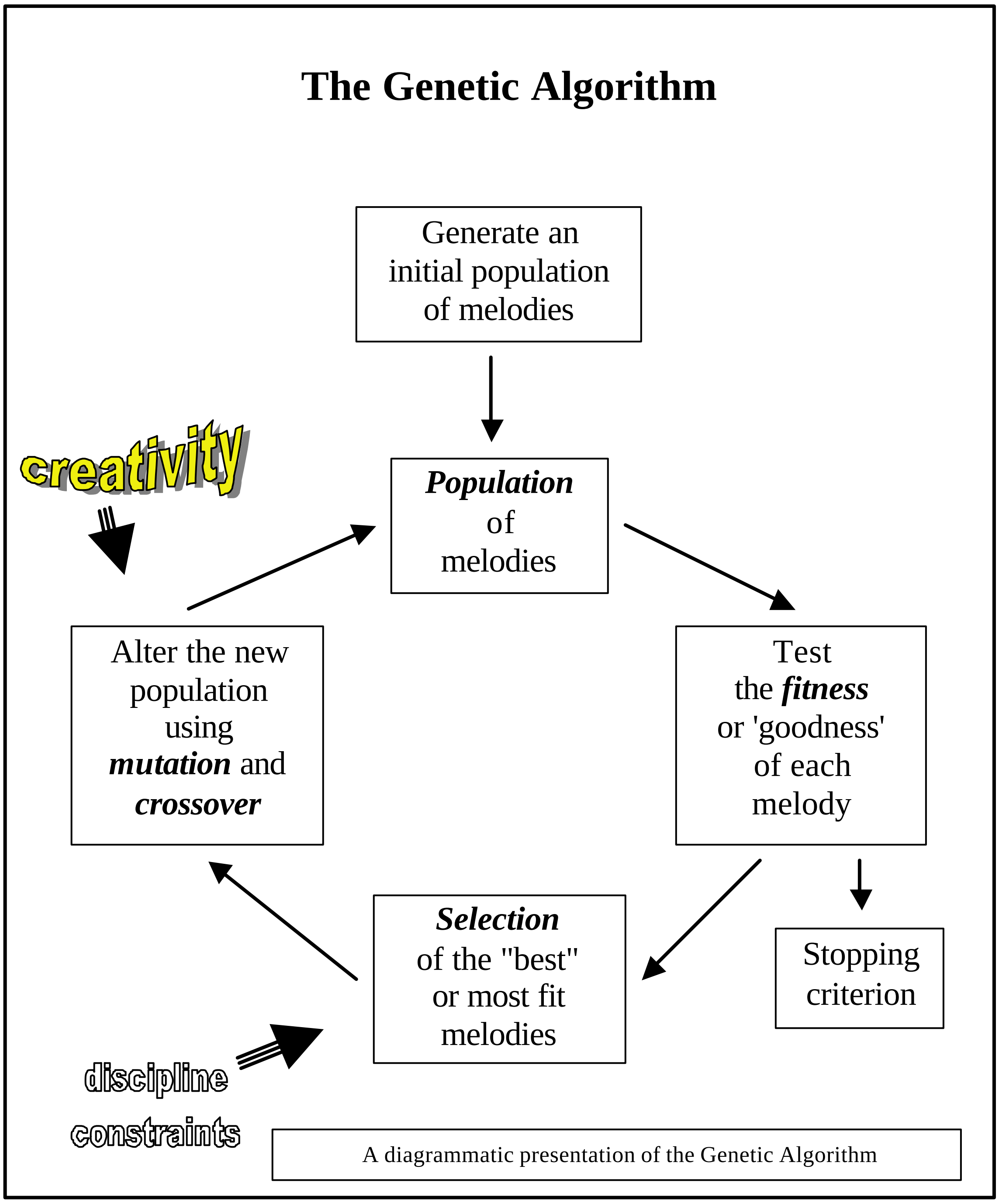
In the language of artistic endeavour, this is the tension between creativity and the discipline of aesthetic and technical constraints. In the language of artificial intelligence and machine learning, it is the trade-off between the ability to explore a large search-space of potential actions and the constraints and bias imposed by the requirement for an efficient search.

Rule-based systems lean strongly to the structure-side of the trade-off [Todd and Werner, 1999]. Their starting point is structure, with novelty (if permitted), included as an add-on. Learning systems lie around the centre of the trade-off because they have some ability to generalise to novel situations within the limits of the statistical knowledge they have acquired. GA's on the other hand have both the ability to explore a large search space (creativity in the guise of the mutation and crossover operators) and constraints or structure (in the guise of a fitness function) explicitly built into the algorithm. The designer has the ability to manipulate the fine balance between the two. This is what makes GA's so appealing for the music composition task.

Considering even the modest task of short (say two bar) melodic extensions, the search space of all possible extensions is still astronomical in size. The success of the GA approach, therefore, depends on imposing appropriate constraints on the search, and, in particular on formulating a fitness function which somehow measures the aesthetic worth of a melody.

Three types of fitness function or *critic* (as the fitness function is sometimes called in the context of music composition) have been explored; the human critic, the rule-based critic and the learning-based critic. Perhaps because of a prejudice that the beauty of music cannot be captured in rules, many attempts have been made to use human experts (musicians) to rate GA composed melodies, say giving them a value out of 10. These methods are defeated by the mental stamina required of humans to objectively assess, possibly thousands of tunes over the course of many GA generations, the great majority of which will not be tuneful.

When, however we turn to rule-based and learning-based critics, we find that they are subject to the same defects as when they are applied directly to the task of music composition. Rules are brittle and neural network learning systems learn only small-scale structure. A genetic algorithm can only produce melodies as good as the knowledge encoded in its critic or fitness function. The conclusion from one decade of research on this problem is that a combination of all three critics is likely to be necessary. For example, one could use a rules and neural networks to weed out the least fit melodies and use human critics to rate just the best. However a report by Biles et al. [1996] expresses dissatisfaction even with this approach.


The Genetic Algorithm
Generate an initial population of melodies
Population of melodies
Test the fitness or 'goodness' of each melody
Stopping criterion
Selection of the "best" or most fit melodies
Alter the new population using mutation and crossover
creativity
discipline
constraints

A diagrammatic presentation of the Genetic Algorithm

This paper considers yet another approach to the problem of constructing a fitness function, by incorporating another component, namely global statistics obtained from the analysis of a library of melodies. We asked the question; "are there any quantitative features or melodic properties that can help to define a good melody?" We approached this question by gleaning text books of melodic writing [Sturman, 1983; Sturman, 1986; Howard, 1990] for heuristics which were then condensed into 21 quantitative features. These features involved pitch, tonality, melodic contour, rhythm and repeated patterns or motifs.

Our list of features is not exhaustive (for example, we have not yet included features describing the harmonic function of note sequences) and our melodic analysis is limited to the world of short melodies in the tradition of Western diatonic music. Some additional restrictions are that; the compositional activity is monophonic, the representational system is common practice notation, rhythms are regular using only simple time signatures and there is no modulation of key. The chosen features are intended to reinforce the music theory ‘rules’ in the school curriculum, rather than being overly ‘novel’ or ‘creative’. Thus the melodic suggestions produced by the GA should be ‘appropriate’ but do not need to be ideal.

The software environment for this project is the Java-based jMusic composition language which is an ongoing research project of the QUT music department [Sorensen and Brown, 2000]. This language provides a music data structure and visualisation interface for notation and MIDI file reading and writing routines for accessing the sample melodies for analysis.

The melodic features

Each of the 21 features is expressed as a dimensionless ratio (taking a value from 0 to 1) which allows for subsequent combining of features, for example into weighted sums. In addition to an English description, each ratio is defined by its numerator and denominator.

In the following, '#' is to read as "the number of". All pitch intervals are measured in semitones. A quantum is a unit representing the shortest duration note that occurs in any melody. For these analyses, the quantum is a semi-quaver.

Pitch Features

(1) Pitch variety
The ratio of distinct pitches to notes. This feature is a measure of the diversity of the pitch class set used in writing the melody.
Numerator = # distinct pitches
Denominator = # notes.

(2) Pitch range
The pitch range divided by the maximum desirable pitch range. This feature provides an indication of the amplitude of the melodic contour.
Numerator = highest pitch - lowest pitch
Denominator = 24.
For our purposes the maximum desirable range was set to two octaves. If numerator exceeded 24, the ratio was set = 1.0.

Tonality Features

(3) Key Centered
The proportion of quanta where the pitch is primary, that is either tonic or dominant. The feature provides an indication of how strongly the melody has a sense of key.
Numerator = # primary pitch quanta
Denominator = # quanta

(4) Non-scale Notes
The proportion of quanta where the pitch is non-scale (i.e. not in the major or harmonic minor scale as appropriate). This feature provides an indication of how strongly tonal the melody is.
Numerator = # non-scale quanta
Denominator = # quanta

(5) Dissonant Intervals
The fraction of dissonant intervals. The dissonance rating of an interval (measured in semitones) is;

Interval	Dissonance rating
0, 1, 2, 3, 4, 5, 7, 8, 9, 12	0.0
10	0.5
6, 11, ≥13	1.0

Rests are ignored in this feature.
Numerator = sum of all interval dissonances
Denominator = # intervals = # notes -1.

Contour Features

(6) Contour Direction
The sum of all rising intervals divided by the (absolute) sum of all intervals. This feature checks the overall tendency of the melody to rise or fall. A melody starting and finishing on the same note will score 0.5 - higher scores indicates an overall rise while lower scores indicate a descending contour. Rests are ignored in this feature.
Numerator = Sum of all rising intervals
Denominator = Sum of all intervals

(7) Contour Stability
The proportion of intervals for which the following interval is in the same direction. This is a measure of stability in melodic direction. Two consecutive intervals incorporate three notes and if all three notes have the same pitch, this is counted as moving in the same direction. Rests are ignored in this feature.
Numerator = # consecutive intervals moving in same direction.
Denominator = # intervals -1

(8) Movement by Step
The proportion of intervals that are diatonic steps. A high score indicates a smooth melodic curve with few large leaps. A diatonic step interval will be one or two semitones. Rests are ignored.
Numerator = # diatonic steps
Denominator = # intervals

(9) Leap Returns
The proportion of large (leap) intervals NOT followed by a return interval. A large leap is greater than or equal to eight semitones (minor 6th). The returning interval must be at least 1 semitone but less than the leap interval preceding it.
Numerator = # large leaps not followed by a return interval
Denominator = # large leap intervals

(10) Climax Strength
Measured as the inverse of the number of times the climatic note is repeated in the melody. The highest value of 1 for this feature occurs when the climatic note is used only once. More frequent use lessens the climatic impact.
Numerator = 1
Denominator = # of uses of climatic note

Rhythmic Features

(11) Note Density
The ratio of notes to quanta. This feature indicates the sparseness versus 'business' of a melody.
Numerator = # notes
Denominator = # quanta

(12) Rest Density
The proportion of silent quanta. This feature indicates the degree of silence or sparseness of a melody.
Numerator = # silent quanta
Denominator = # quanta

(13) Rhythmic Variety
This feature measures the degree of use of the 16 rhythmic values (note durations) between semi-quaver and semi-breve. It gives some indication of the rhythmic coherence of a melody.
Numerator = # distinct note durations used
Denominator = 16

(14) Rhythmic Range
The range of rhythmic values in the melody divided by the maximum possible range. The maximum range = 16 (i.e. semi-breve ÷ semi-quaver). This feature is different from the previous one, *rhythmic variety*, in that it is possible for a melody to have a wide range of note durations but not use intermediate values.
Numerator = Max note duration ÷ min note duration
Denominator = 16

(15) Syncopation
The proportion of notes that are syncopated. That is, the proportion of notes having duration ≥ one beat and which start off the beat. Rests are differentiated from notes in this feature.
Numerator = # syncopated notes
Denominator = # notes

Patterns

(16) Repeated pitch
The proportion of intervals where both notes are of same pitch. It is another measure of melodic movement (or the lack of it). For this feature, rests are ignored i.e., it considers all note to next note pairs.
Numerator = # repeated pitch intervals
Denominator = # intervals

(17) Repeated rhythmic value
The proportion of intervals where both notes have the same duration. Rests are ignored i.e., it considers all note to next note pairs.
Numerator = # repeated duration intervals
Denominator = # intervals

(18) Repeated Pitch Patterns of 3 notes
The proportion of three note sequences whose pitch sequence is subsequently repeated. Rests are taken into account such that they may be part of a pattern.
Numerator = # repeated 3 note pitch sequences
Denominator = # notes - 4

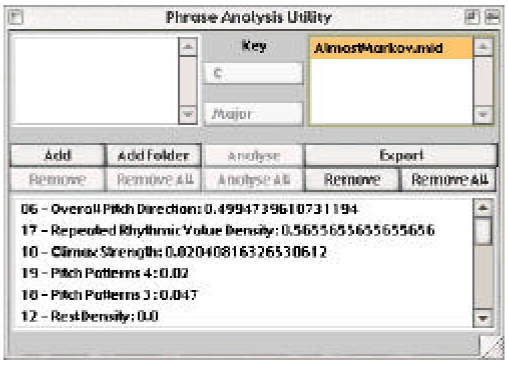
(19) Repeated Pitch Patterns of 4 notes
The proportion of four note sequences whose pitch sequence is subsequently repeated. Like (18) this feature compares intervalic relationships. Rests are taken into account such that they may be part of a pattern.
Numerator = # repeated 4 note pitch sequences
Denominator = # notes - 5

(20) Repeated Rhythm Patterns of 3 notes
The proportion of three note sequences whose note duration sequence is subsequently repeated. This feature measures the rhythmic coherence of the melody as developed through repetition. The rhythmic ratio is checked rather than absolute rhythms, therefore, a pattern crotchet, minim, crotchet will match quaver, crotchet, quaver. Rests are taken into account such that they may be part of a pattern.
Numerator = # repeated 3 note-duration sequences
Denominator = # notes - 4

(21) Repeated Rhythm Patterns of 4 notes
The proportion of four note sequences whose note-duration sequence is subsequently repeated. This feature measures the rhythmic coherence of the melody as developed through repetition. Like feature (20) the rhythmic ratio is checked rather than absolute note durations. Rests are taken into account such that they may be part of a pattern.
Numerator = # repeated 4 note-duration sequences
Denominator = # notes - 5

Methods

The analysis of features used MIDI files as a readily available source of melodic data. Code was written in the Java language, using the jMusic MIDI reading and writing libraries, to handle the analysis process. The interface for the analysis program is shown in figure 1. Melodies from the Western art music repertoire were sourced as MIDI files and were appropriately prepared for analysis by quantising rhythms, transposing into C major or A minor, and setting playback to 120 beats per minute. This normalised the data such that reasonable comparisons could be drawn between them.



```
06 - Overall Pitch Direction: 0.49947396107311194
17 - Repeated Rhythmic Value Density: 0.5655655655655656
10 - Climax Strength: 0.02040816326530612
19 - Pitch Patterns 4 : 0.02
18 - Pitch Patterns 3 : 0.047
12 - Rest Density: 0.0
```

Figure 2. Interface for the Melodic Analysis Application

Our library currently includes 36 melodies, including 18 classical melodies by Bach, Mozart, Beethoven and Tchaikovsky; 10 pre-classical compositions by Du Fay, Gesualdo, Gibbons, Montiverdi and Palestrina; six traditional nursery rhymes and two melodies from popular tunes. Note that this is a preliminary report and this small number of melodies will be expanded as the project proceeds.

The 21 features were calculated for each melody and the average value and standard deviation of each feature was calculated for the whole group and various style groups, as described below.

Results and Discussion

Our first interest was to look for features which would place a strong constraint on the 'fitness' of a melody, that is features whose values have a small standard deviation. Six features having the lowest standard deviations are listed in Table 1.

Feature	Value
1. Pitch Variety	0.27 ± 0.11
5. Dissonant Intervals	0.01 ± 0.02
6. Contour direction	0.49 ± 0.06
7. Contour Stability	0.40 ± 0.11
13. Rhythmic Variety	0.24 ± 0.07
14. Rhythmic Range	0.32 ± 0.11

Table 1.

The means and standard deviations of the six melodic features having lowest standard deviations

A value of 0.27 for Pitch Variety means that *on average* each pitch is repeated 3.7 times in a melody. This value is surprisingly constant across all melodic styles. Dissonant intervals (as defined by feature 5) appear very infrequently in all melodic styles. Contour direction is always close to 0.5 (feature 6), (that is, melodies tend to start and finish close to the same note) and changes in melodic direction occur on average every 1.6 intervals. Across all styles of melody, only 1/4 of the available note durations is used (feature 13), that is, semi-quaver, quaver, crotchet and minim. This restricted selection is the result of excluding compound time melodies from our library.

It was surprising to observe the degree to which some commonly taught heuristics were not observed in our library of melodies. For example, the rule to avoid repetition of the climatic note was violated observed in 21 of the 36 melodies. In two nursery rhymes the climatic note was repeated 11 times! One out of 5 large leaps was not followed by a return and this rule was not observed in 8 of the melodies.

Feature	Melodic Style		
	Classical	Early	Nursery
2. Pitch range	0.56 ±0.17	0.50 ±0.07	0.45 ±0.07
4. Non-scale notes	0.19 ±0.24	0.19 ±0.24	**0.00 ±0.01
8. Step movement	0.62 ±0.26	0.62 ±0.32	*0.91 ±0.06
11. Note density	**0.61 ±0.20	**0.38 ±0.10	0.48 ±0.18
17. Repeated duration	*0.66 ±0.19	*0.44 ±0.22	0.51 ±0.12
18. Repeated pitch patterns of 3 notes	0.03 (11/18)	0.06 (5/9)	0.11 (7/7)
19. Repeated pitch patterns of 4 notes	0.02 (8/18)	0.03 (4/9)	0.08 (7/7)
20. Repeated duration patterns of 3 notes	0.34	0.19	0.20
21. Repeated duration patterns of 4 notes	0.26	0.11	0.14

Table 2.

The means and standard deviations of features that distinguished melodic style. * indicates the means are significantly different at 1%; ** at 0.5%.

A number of features were useful in discriminating melodic styles (Table 2). Pitch Range (Feature 2) was close to an octave for all three categories, slightly exceeding the octave for classical compositions and slightly under for nursery rhymes. But of greater interest is that the variation in range for nursery rhymes and early music is very small. Non-scale notes (Feature 4) are almost non-existent in nursery rhymes whereas for the other two style 20% of notes on average do not belong to the scale. Nursery rhymes are also distinguished by 90% of intervals being stepwise.

Early music melodies have a significantly lower note density than classical (feature 11) and a significantly lower proportion of repeated note durations (feature 17). These two features are probably not independent.

By our definition, features 18 and 19 (repeated 3 and 4 note sequences of pitch) have very low values. (SD's for features 18-21 were large compared to means and have been excluded for clarity.) However if we count melodies in which a repeated pattern occurs (shown as a fraction in Table 2), we find that all six nursery rhymes displayed 4 note repeated sequences and 11 of the 18 classical melodies displayed 3 note sequences. Rhythmic patterns (i.e. features 20 and 21) occur with a much higher frequency than pitch patterns and are most frequent in classical melodies.

Having explored the features in isolation, the question arises as to interactions between the features. Do some features consistently occur together? Or put another way, do the melodies cluster in feature space?

Using PCA to optimally project the feature vectors onto a two dimensional surface (see Figure 2 in which each numbered point represents a single melody), we do not observe any obvious clustering. The four principle components of the first and second PCA axes and their means and standard deviations are as follows;

PCA Axis 1

(4) non-scale notes	0.16 ±0.23
(8) step movement	0.67 ±0.28
(9) leap returns	0.19 ±0.38
(10) climax strength	0.61 ±0.35

PCA Axis 2

(11) note density	0.51 ±0.20
(17) repeated duration	0.57 ±0.20
(20) 3 note dur. pattern.	0.26 ±0.24
(21) 4 note dur. pattern.	0.19 ±0.23

Not surprisingly, these are the eight features having the highest standard deviations. Furthermore, all except features (9) and (10) appear in Table 2 as features which can be used to distinguish melodic style.

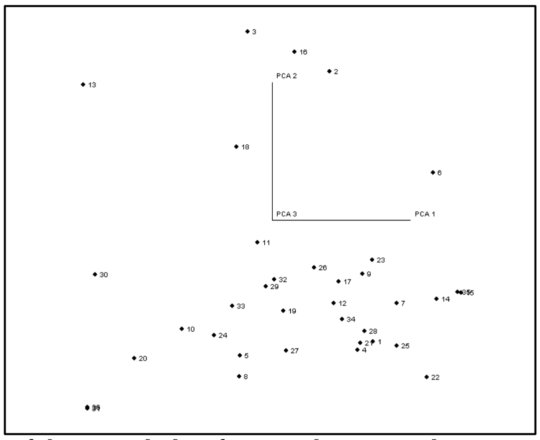


Figure 2. A PCA projection of the 36 melodies from 21 dimensional space to two dimensions. For ease of subsequent interpretation, note that a third dimension exists perpendicular to the page, i.e. the image has depth.

Although no obvious clusters appear in Figure 2, we can subject the 36 points to a clustering algorithm (we used Forgy's variant of Lloyd's vector quantisation algorithm) and superimpose those clusters on the PCA projection. An important parameter to be determined is the appropriate number of clusters. We graphed the quantisation error over a range of cluster numbers, and observed an inflection at 9 clusters (Figure 3), which suggests this as a 'natural' cluster number. Figure 4 illustrates these nine clusters superimposed onto the PCA image. Although Forgy's algorithm establishes spherical clusters, some have been shown elliptically in Figure 4 for clarity. Note that the clusters do not in fact overlap. Rather they are separated in the third or depth dimension, perpendicular to the page.

Clusters 1, 5 and 6 contain only classical melodies distinguished in particular by a high value for the note density feature. However in general, the larger clusters, 3, 4 and 7, include melodies of all styles.

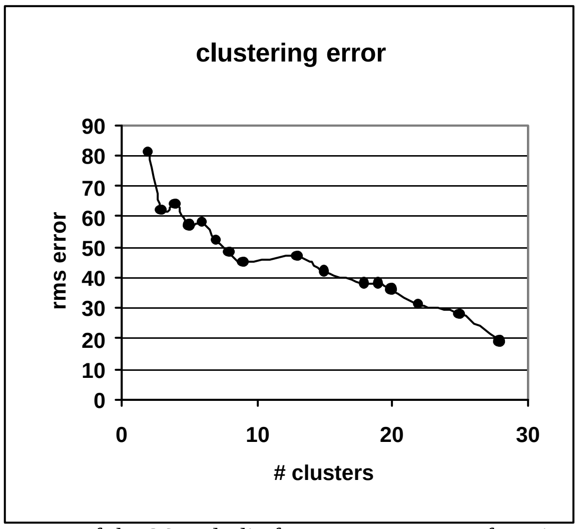


Figure 3. Quantisation error of the 36 melodic feature vectors as a function of number of clusters

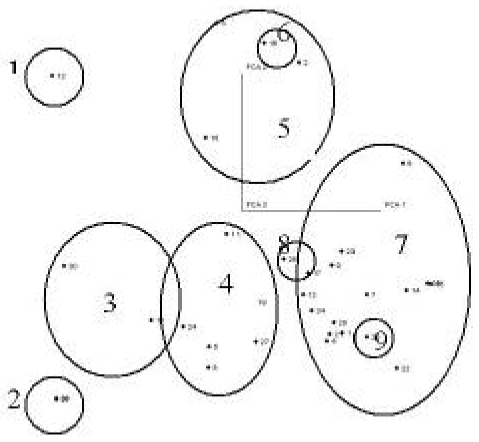


Figure 4. The superimposition of the 9 spherical clusters obtained by vector quantisation onto the PCA projection of the 36 melodies

We also used a second clustering algorithm which works well if the data tend to be clustered elliptically rather than spherically. The appropriate number of elliptical clusters was five (see Figure 5). Clusters 1 and 5 have merged with clusters 3 and 4 respectively and clusters 6, 8 and 9 and merged with parts of cluster 7. The clusters tend to be elongated along the direction of the second PCA axis which is a nice confirmation that the features of the second PCA axis are less important in clustering the melodies than those of the first PCA axis.

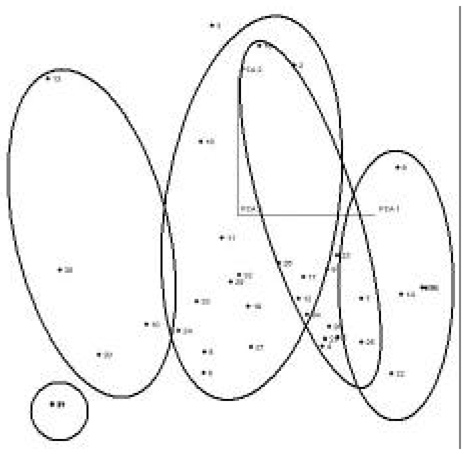


Figure 5. The superimposition of 5 elliptical clusters obtained by a modified VQ algorithm onto the PCA projection of the 36 melodies

Conclusion

The (normalised) quantitative melodic features we have described above were developed based on rules of best practice in melodic writing as taught to students of melody. Some of the rules are vague heuristics at best, such as, there should be variety of pitch, rhythm and contour. Analysis of our library of melodies has helped to put quite strong quantitative constraints on some of these vague heuristics and this is helpful in the attempt to construct a GA fitness function. For example, melodies whose values for the features shown in Table 1 are more than two standard deviations from the mean, would receive a low fitness rating.

Some quite specific heuristics, such as the leap return rule and the repeated climax rule were often found to be violated. This was particularly the case with nursery rhymes. It could be a conclusion of this work that nursery rhymes should be excluded from our library because they exhibit features (such as repeated climax and little contour variation) that are not necessarily best practice for students.

It is not possible to draw strong conclusions from the cluster analysis of our library of melodies because 36 melodies is too few in number. It remains our hope, that given sufficient melodies, we should be able to observe clusters in feature space. Candidate melodies falling in sparsely populated parts of feature space would then be given a low fitness rating.

It is not our intention that a GA fitness function could be composed entirely of features determined by global statistics. As already indicated, the conclusion of the past decade of research is that a melodic fitness function needs to include a variety of components. The melodic statistics we have outlined in this paper represent another addition. Of great interest is the possibility that the aesthetic appeal of a melody can be understood partly in terms of auditory neurology [Katz, 1994]. To the extent that melodic content could be mapped to a quantitative neurological response, this would represent a powerful addition to the fitness measurement.

Finally, the melodic statistics described in this paper have more uses than just for inclusion in a fitness function. Both the initialisation of the first population and the mutation and cross-over operators can be biased towards a melodic norm or even a stylistic norm if the appropriate library of melodies is chosen. It is becoming clear that to render GA methods effective for musical activity, the incorporation of domain specific knowledge in the form of *knowledge intensive operators*, is not just desirable but a necessity [Wiggins et al., 1998].

Acknowledgments

We acknowledge receipt of a Queensland University of Technology Special Small Research Grant. Thanks to Adam Kirby for Java programming using jMusic. An early version of this paper was published in the ISSEI2000 CD-ROM proceedings of the First International Workshop "Developing Creativity and Large Mental Outlook in the Computer Age". [Towsey, Brown, Wright and Diederich, 2000].

References

Biles, J. A. (1994). GenJam: A genetic algorithm for generating jazz solos. *ICMC'94 Proceeding*, The International Computer Music Association.

Biles, J. A., Anderson, P. G. & Loggi, L. W. (1996). Neural network fitness functions for a musical GA. *Proceedings of the International ICSC Symposium on Intelligent Industrial Automation (ILA'96) and Soft Computing (SOCO'96),* Reading, UK: ICSC Academic Press, B39-B44.

Burton, A. R. & Vladimirova, T. (1999). Generation of Musical Sequences with Genetic Techniques. *Computer Music Journal*, 23 (4), 59-73.

Cope, D. (1997). The Composers Underscoring Environment: *CUE. Computer Music Journal*, 21 (3), 20-37.

Ebcioglu, K. (1984). An expert system for Schenkerian synthesis of chorals in the style of J.S. Bach. *Proceedings of the 1984 International Computer Music Conference*, San Francisco: International Computer Music Association, 135-142.

Horner, A. & Ayers, L. (1995). Harmon-isation of musical progression with genetic algorithms. *ICMC'95 Proceedings*, San Francisco: International Computer Music Association, 483-484.

Horner, A. & Goldberg, D. E. (1991). Genetic algorithms and Computer Assisted Music Composition. *ICMC'91 Proceedings*, San Francisco: International Computer Music Association, 479-482.

Howard, J. (1990). *Learning to Compose*, UK: Cambridge University Press.

Katz, B. F. (1994). An Ear for Melody. *Connection Science*, 6, 299-324.

Laske, O. E. (1992). AI and Music: A cornerstone of cognitive musicology. In Balban, M., Ebcioglu, K. & Laske, O. (Eds.) *Understanding Music with AI: Perspectives on music cognition*, Menlo Park, CA: The AAAI Press, 3-29.

Loy, D. G. (1989). Composing with computers. In Mathews, M.V. & Pierce, J.R. (Eds.), *Current Directions in Computer Music Research*. Cambridge, MA: MIT Press, 20-36.

Loy, D.G. (1991) Connectionism and Musiconomy. In P. M. Todd & D. G. Loy (Eds.) *Music and Connectionism*, Cambridge, MA: MIT Press, 20-36.

Moorer, J. A. (1972). Music and Computer Composition. Reprinted in Schwanauer, S. & Levitt, D. (1993) *Machine Models of Music*, Cambridge, MA: The MIT Press, 167-188.

McIntyre, R. A. (1995). Composition with genetic algorithms. *Technical report*, University of Michigan.

Sorensen, A. & Brown, A. (2000). Introducing jMusic. *Paper presented at the Australasian Computer Music Conference (interFACES)*, 5-8 July, 2000, QUT, Brisbane.

Sturman, P. (1983). *Harmony, Melody and Composition*, UK: Longman Group.

Sturman, P. (1986). *Advanced Harmony, Melody and Composition*, UK: Longman Group.

Todd, P. M. & Werner, G. M. (1999). Frankensteinian Methods for Evolutionary Music Composition. In Griffith and Todd, P. M. (Eds.) *Musical networks: Parallel perception and performance*, 313-339.

Towsey, M., Brown, A., Wright, S., & Diederich, J. (2000). Interactive Music Composition using Genetic Algorithms. *Paper presented at the 7th Conf. of the International Society for the Study of European Ideas (ISSEI)*, 14-18 August, 2000, University of Bergen, Norway.

Wiggins, G., Papadopoulos, G., Phon-Amnuaisuk, S. & Tuson, A. (1998). *Evolutionary methods for musical composition*,
http://www.dai.ed.ac.uk/daidb/papers/documents/rp882.html.