Evolving Viable Pitch Contours

Kristian Guillaumier
kguil@cs.um.edu.mt
Dept. of Computer Science and AI
University of Malta

ABSTRACT

At a very basic level, a piece of music can be defined as an organised arrangement of sounds[1] occurring both sequentially (as in melody) and concurrently (as in harmony). As music evolved into a science and an established form of art, people started studying the characteristics of these sounds and drew sets of guidelines and rules, that if followed would produce pieces of music that are aesthetically more pleasing than others. Early examples can be seen in Pythagoras' observations and experiments with blacksmiths' hammers [1]. Allegedly some 2500 years ago, he was walking by a blacksmith's shop when he heard the ringing tones of hammers hitting an anvil. Upon further observation, he realised that a hammer weighing half as much as a previous one sounded twice as high in pitch (an octave – ratio 2:1). A pair of hammers whose weights had a ratio of 3:2 sounded a fifth apart. Eventually he came to the conclusion that simple ratios sounded good.

In this paper, we are concerned with the generation of musical phrases constrained by the rules that governed music developed during the so called Common Practice Period (CPP). This period refers to an era in musical history spanning from the 17th to the early 20th centuries [2] and included the Baroque and Romantic styles amongst others. Colloquially, music in the style of the CPP is sometimes better (but incorrectly) known as 'Classical' music.

General Terms

Aleatoric composition, music theory, common practice period, genetic algorithms.

1. INTRODUCTION

Computers have been used as an aid in composing music since the mid-1950s and the techniques generally employed fall into the categories of aleatoric composition and processes that return permutations of predefined musical elements such as pre-composed measures of music [6]. In the former technique, stochastic methods are used to generate sounds – possibly utilising some musical observations to guide random processes. In the latter, a number of predefined measures of music are selected and attached to each other to yield a piece of music. This technique has been practiced since the 18th century using an algorithm known as the *Musikalisches Würfelspiel* (musical dice game) [7]. Mozart has been known to compose a number of Minuets based on this algorithm. An interactive example of this method can be found at [8].

In this paper, we present a Genetic Algorithm (GA) designed to generate pitch contours[2] that conform to the rules used in the CPP. Clearly, the rules we will be considering form the basis of the fitness function of the GA. Here we assume that the reader is familiar with basic–to–intermediate theory of music. We refer beginners to any introductory textbook on the matter such as [3], [4] and [5].

1.1 Rules for Developing Melodies/Pitch Contours

1. Notes in the melody should be diatonic.
2. Most of the melody must progress in stepwise motion.
3. The melody should contain a number of leaps. The number of leaps depends on the length of the melody and a leap must always occur in the context of contrary motion. Also, leaps must be a consonant interval.
4. Melodies should cover the whole range of notes assigned to it.
5. The highest note (climax) in a melody should occur only once, usually in the latter half. This climax note should be melodically consonant with the tonic.
6. Certain note intervals such as augmented intervals are unacceptable.
   (a) Augmented intervals are not allowed.

[1]In this context we refer to sounds of a musical nature – notes.

[2]A pitch contour is a sequence of notes that sound melodical but without possessing any rhythm (essentially, a melody composed with notes of a single duration only such as crotchets). Melodies are rhythmically more complex than pitch contours, but in this text the terms pitch contour and melody are synonymous.

   (b) A diminished interval is not allowed unless the note following the interval falls between that interval by a perfect or imperfect interval.

7. The note on the seventh degree of the scale must rise stepwise to the tonic.
8. Melodies in a major scale should start and end with the tonic or dominant. Melodies in a minor scale should start and end with the tonic or mediant.
9. Only notes in a single soprano, alto, tenor or bass voice should be used. Arbitrarily, the soprano range is considered here (from middle C to G two octaves up).
10. Notes should be repeated rarely.

2. THE GENETIC ALGORITHM

Creating a melody conforming to a set of rules can be construed as a constraint satisfaction problem. Consider an 8-note melody to be composed using any of 7 notes (an octave-worth of diatonic notes). These parameters would yield 5,764,801 (7^8) different melodies – the search space. Longer, more varied melodies would clearly increase the search space immensely making the problem non-trivial. In this section we describe a GA used to search the space for melodies that conform to the rules outlined earlier on. It is assumed that the reader is familiar with the mechanics of a GA and its theory. Readers are referred to [9] for a thorough exposition.

2.1 Chromosome Structure

The scheme used to represent a candidate pitch contour as a chromosome in our algorithm is straightforward. The chromosome is an array of integers, where each integer represents a note value in the contour. The integer values representing each note are borrowed from the Musical Instrument Digital Interface (MIDI) standard, where, for example, middle-C is represented by the value 60, C# by the value 61, D by the value 62, etc... This idea is illustrated in Figure 1.

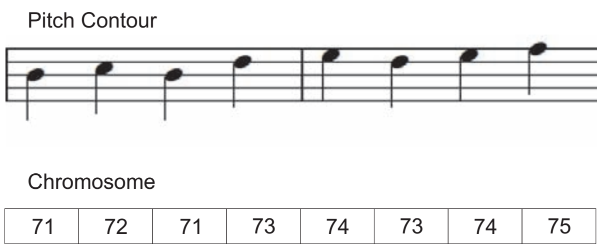

Figure 1: Chromosome representation of a pitch contour.

The simplicity of this representation scheme allows for uncomplicated designs for crossover and mutation operators. Additionally, the fact that the note values map to the MIDI standard allows us to export the contour as a MIDI file for quick auditioning or editing purposes.

2.1.1 *Pitch Sets*

The possible note values for each locus in the chromosome (i.e. each possible note in the contour) are selected from a set of permissable notes called the *pitch set*. Essentially the pitch set defines which notes the melody can be composed of. Restricting the melody to use only notes from a pitch set has a number of advantages:

1. Ensuring that the notes in the pitch set are diatonic, implies that any candidate pitch contour will be diatonic as well. This implicitly enforces the first rule mentioned previously.
2. Similarly, by ensuring that the notes in the pitch set are within the range of a single voice (e.g. the soprano voice), rule 9 above is implicitly observed.
3. The search space is reduced to the various permutations of the notes in the pitch set rather than all the notes in the range of a particular instrument.

2.1.2 *Fixed Notes*

During setup of the algorithm, certain loci in the chromosome can be fixed to certain notes. For example, the algorithm can be instructed that the first note in the melody should always be middle-C and no operator, such as a mutation, would be allowed to change it. The allows us to ensure that the pitch contour would, for example, always start with the tonic and end with the dominant. By specifying fixed notes in 'the middle' of the melody we can give it a particular texture or shape. Additionaly, since most notes are required to progress in stepwise motion, the fixed note effectively becomes an attractor for other notes thereby serving as a climax note.

2.1.3 *Initialisation*

In the initial population, chromosomes are initialised to pitch contours with notes randomly selected from the pitch set. Fixed notes in the pitch contour are observed – the fixed notes in any pitch contour are immutable.

2.2 The Fitness Function

The fitness function developed is penalty-based. A faultless melody has a fitness value of zero whilst the fitness of a flawed melody is negative. Essentially, for each rule that is violated, a penalty value is deducted from the fitness. The penalties for each rule are interpreted as follows:

1. RULE: Notes in the melody should be diatonic.
   INTERPRETATION: This rule can never be violated because a chromosome can only be composed of notes selected from a pitch set whose notes are already guaranteed to be diatonic. This rule is implicitly observed.
2. RULE: Approximately n% of the melody must progress in stepwise motion.
   INTERPRETATION: Determine the number of expected stepwise intervals in the melody from n. By observing the actual intervals in the candidate melody, determine the number of actual stepwise intervals. If the actual number of stepwise intervals is less than expected, apply a penalty (e.g. -5) to the fitness for each expected interval that is not present.
3. RULE: The melody should contain a number of leaps. The number of leaps depends on the length of the melody and a leap must always occur in the context of contrary motion. Also, leaps must be a consonant interval.
   INTERPRETATION: Determine the number of leaps in

the melody. If the actual number of leaps differs from some required amount, penalise in proportion to this difference. If the leap is not consonant, apply a penalty. For contrary motion, if the leap is preceded by a note that is not within its interval, apply a penalty. Finally, if the leap is followed by a note that is not within its interval, apply a penalty too.

4. Rule: Melodies should cover the whole range of notes assigned to it.
   Interpretation: For each note in the pitch set to be used that does not occur in the melody, apply a penalty.

5. Rule: The highest note (climax) in a melody should occur only once, usually in the latter half. This climax note should be melodically consonant with the tonic.
   Interpretation: Let c be the the number of times the highest note in the melody occurs. Apply a penalty to the fitness (c-1) times. The requirement of the climax note occurring in the latter half of the melody and being consonant to the tonic can be implicity observed by setting the climax note as a fixed note in the chromosome.

6. (a) Rule: Augmented intervals are not allowed.
   Interpretation: Apply a penalty for each augmented interval in the melody.
   (b) Rule: A diminished interval is not allowed unless the note following the interval falls between that interval by a perfect or imperfect interval.
   Interpretation: For each diminished interval in the melody, if the next note does not lie between that interval and is a perfect or imperfect interval, apply a penalty. Also, a penalty is applied if the melody ends in a diminished interval (there would not be a note following the interval).

7. Rule: The note on the seventh degree of the scale must rise stepwise to the tonic.
   Interpretation: For each note on the seventh degree of the scale that does not rise to the tonic, apply a penalty.

8. Rule: Melodies in a major scale should start and end with the tonic or dominant. Melodies in a minor scale should start and end with the tonic or mediant.
   Interpretation: This rule is implicitly observed by setting the starting and end notes as fixed notes in the chromosome.

9. Rule: Only notes in a single soprano, alto, tenor or bass voice should be used. The soprano range is considered here (from middle C to G two octaves up).
   Interpretation: This rule is implicitly observed by setting the notes in the allowed pitch set to those in a desired voice range.

10. Rule: Notes should be repeated rarely.
    Interpretation: Count the number of repeated notes in the melody. If this amount varies from some expected value, apply a penalty proportional to this difference.

2.3 Genetic Operators and Other Parameters

2.3.1 Crossover

In our algorithm, the crossover operator is a basic implementation of single-point crossover. A illustrated in Figure 2, a common locus in two parent melodies is randomly selected, and notes between the two parents are swapped along that interval. Parent melodies are selected to participate in crossover using a roulette-wheel selection scheme.

A *crossover rate* value governs how many parents are selected for crossover purposes. A crossover rate of 80% means that after selection, 80% of the new population will be made up of offspring that were generated by mating the parents. The remaining 20% of the new population is populated with new, randomly-created chromosomes.

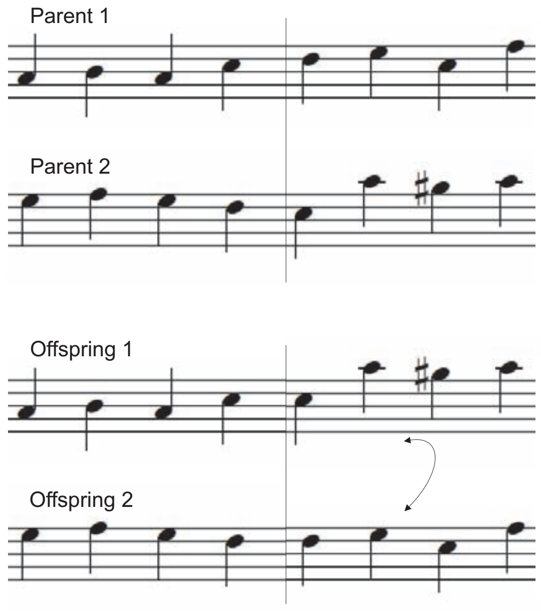

Figure 2: Single-Point Crossover between two pitch contours.

2.3.2 Random Mutations

In the algorithm a random mutation is implemented as:

1. The replacement of a random note in the melody with any random one in the pitch set of allowed notes.

2. The swapping of two, random notes in the melody.

A *random mutation rate* is used to determine the probability that a child chromosome created after crossover will be randomly mutated. The mutation operator is partial to the fixed note configuration of the chromosomes. A mutation is aborted if it would effect a fixed note.

2.3.3 Guided Mutations

This operator, works by altering a note in an attempt to rectify a deficiency in the melody. For example, if it is determined that a melody contains too little stepwise motion, a note in a non-stepwise interval is changed to make that interval stepwise. Similarly, if an augmented interval is found in a melody, a note in that interval is replaced to change the improper interval.

A *guided mutation rate* is used to determine the probability that a random child chromosome created after crossover will be mutated 'intelligently' as described above.

2.3.4 *Elitism*

The roulette–wheel selection mechanism used ensures that the fittest parents are paired to yield the new offspring pitch contours. Nonetheless, there is always the risk that the pitch contours generated after crossover and possible mutations have a fitness less then that of the original parent contours that spawned them. This implies that there is a possibility that, over time, the overall fitness of the population could degrade. To avoid this risk, an *elitism rate* parameter is used. This parameter represents a percentage of the best contours in the current population that will replace random contours in the next one.

3. EXPERIMENTAL RESULTS

In this section, we present a number of experimental runs of the algorithm and their respective results.

3.1 General Notes

1. Pitch contours have been generated in C Major, G Major, A Minor (harmonic[3]) and E Minor (harmonic). In this section, we only present the results of the algorithm when instructed to compose contours in A minor. This choice is arbitrary since setting a different scale for the algorithm to work in will not effect its performance in any way.

2. Various settings for population size, crossover and mutation rates have been used to determine how the GA converges under these conditions.

3. All pitch contours generated have been set to be 16 and 32 notes long.

4. The algorithm converges to multiple optimal solutions in all cases. The only cases observed when an optimal solution could not found were when they either did not exist or the parameters of the GA conflicted. For example when the pitch set is so limited that it would be impossible to satisfy the rate of stepwise motion desired – if the only notes in the pitch set are C5 and G5, it is clearly impossible to move in stepwise motion.

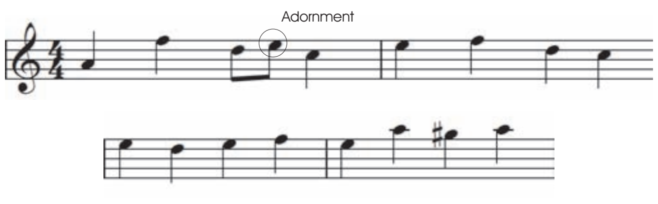

Figure 3: An optimal pitch contour (with an added adornment) generated by the algorithm.

[3]Natural minor with a raised seventh.

3.2 Experiment 1

- Key: A Minor (harmonic).
- Contour Length: 16 notes.
- Pitch Set: {A4, B4, C5, D5, E5, F5, G#5, A5}.
- Note Usage: 80%.
- Fixed Notes: {Pos 1=A4, Pos 15=G#5, Pos 16=A5}.
- Population Size: 100.
- Maximum Number of Generations: 50.
- Crossover Rate: 90%.
- Random Mutation Rate: 5%.
- Guided Mutation Rate: 5%.
- Stepwise motion rate: 80%.
- Leap Rate: 15%.
- Average Generations to Yield Optimal Contours: 25.
- Average Number of Optimal Contours Yielded in Final Generation: 3.
- Example result: A4, B4, A4, F5, E5, C5, B4, C5, D5, C5, D5, E5, D5, A5, G#5, A5.

3.3 Experiment 2

- Key: A Minor (harmonic).
- Contour Length: 16 notes.
- Pitch Set: {A4, B4, C5, D5, E5, F5, G#5, A5}.
- Note Usage: 80%.
- Fixed Notes: {Pos 1=A4, Pos 15=G#5, Pos 16=A5}.
- Population Size: 100.
- Maximum Number of Generations: 50.
- Crossover Rate: 50%.
- Random Mutation Rate: 20%.
- Guided Mutation Rate: 20%.
- Stepwise motion rate: 80%.
- Leap Rate: 15%.
- Average Generations to Yield Optimal Contours: ¿ 50.
- Average Number of Optimal Contours Yielded in Final Generation: 0.2.
- Example result: A4, E5, D5, C5, D5, E5, D5, E5, F5, C5, D5, E5, F5, E5, G#5, A5.

3.4 Experiment 3

- Key: A Minor (harmonic).
- Contour Length: 16 notes.
- Pitch Set: {A4, B4, C5, D5, E5, F5, G#5, A5}.
- Note Usage: 80%.
- Fixed Notes: {Pos 1=A4, Pos 15=G#5, Pos 16=A5}.
- Population Size: 100.
- Maximum Number of Generations: 50.
- Crossover Rate: 100%.
- Random Mutation Rate: 5%.
- Guided Mutation Rate: 5%.
- Stepwise motion rate: 80%.
- Leap Rate: 15%.
- Average Generations to Yield Optimal Contours: 20.
- Average Number of Optimal Contours Yielded in Final Generation: 3.
- Example result: A4, E5, D5, C5, D5, C5, E5, F5, E5, F5, E5, F5, E5, A5, G#5, A5.

3.5 Experiment 4

- Key: A Minor (harmonic).
- Contour Length: 16 notes.
- Pitch Set: {A4, B4, C5, D5, E5, F5, G#5, A5}.
- Note Usage: 80%.
- Fixed Notes: {Pos 1=A4, Pos 25=E5, Pos 32=A5}.
- Population Size: 200.
- Maximum Number of Generations: 200.
- Crossover Rate: 90%.
- Random Mutation Rate: 5%.
- Guided Mutation Rate: 5%.
- Stepwise motion rate: 80%.
- Leap Rate: 15%.
- Average Generations to Yield Optimal Contours: 70.
- Average Number of Optimal Contours Yielded in Final Generation: 15.
- Example result: A4, B4, C5, D5, E5, A4, B4, A4, E5, D5, A5, G#5, A5, C5, D5, C5, D5, C5, B4, C5, E5, D5, E5, D5, E5, F5, E5, D5, E5, F5, E5, A5.

3.6 Experiment 5

- Key: A Minor (harmonic).
- Contour Length: 16 notes.
- Pitch Set: {A4, B4, C5, D5, E5, F5, G#5, A5}.
- Note Usage: 80%.
- Fixed Notes: {Pos 1=A4, Pos 25=E5, Pos 32=A5}.
- Population Size: 200.
- Maximum Number of Generations: 200.
- Crossover Rate: 50%.
- Random Mutation Rate: 20%.
- Guided Mutation Rate: 20%.
- Stepwise motion rate: 80%.
- Leap Rate: 15%.
- Average Generations to Yield Optimal Contours: ¿ 200.
- Average Number of Optimal Contours Yielded in Final Generation: 0.1.
- Example result: A4, B4, C5, D5, E5, A4, B4, A4, E5, D5, A5, G#5, A5, C5, D5, C5, D5, C5, B4, C5, E5, D5, E5, D5, E5, F5, E5, D5, E5, F5, E5, A5.

3.7 Experiment 6

- Key: A Minor (harmonic).
- Contour Length: 16 notes.
- Pitch Set: {A4, B4, C5, D5, E5, F5, G#5, A5}.
- Note Usage: 80%.
- Fixed Notes: {Pos 1=A4, Pos 15=G#5, Pos 16=A5}.
- Population Size: 200.
- Maximum Number of Generations: 200.
- Crossover Rate: 100%.
- Random Mutation Rate: 5%.
- Guided Mutation Rate: 5%.
- Stepwise motion rate: 80%.
- Leap Rate: 15%.
- Average Generations to Yield Optimal Contours: 50.
- Average Number of Optimal Contours Yielded in Final Generation: 25.
- Example result: A4, B4, C5, D5, F5, E5, D5, C5, B4, A4, C5, B4, D5, A4, C5, B4, E5, D5, E5, F5, C5, D5, C5, F5, E5, F5, E5, D5, E5, D5, C5, A5.

3.8 Some Observations

1. As expected, searches for longer contours require a larger population and in some cases more generations to produce a result. This can be easily correlated to the immensely larger search space.

2. Low crossover rate values (albeit higher mutation rates) hinder a successful search for optimal solutions. As the crossover rate decreases, the GA effectively degenerates into a random search with is inadequate for such large search spaces.

3. Very long optimal pitch contours can be discovered. Optimal 64, 96 and 128–note pitch contours could be found using the exact same parameters used in experiment 4 above with the exception that for 128–note melodies, more generations and a slightly larger population size was required for the algorithm to converge to one or more optimal solutions.

4. CONCLUSION

In the title of this paper we labeled the pitch contours we sought to generate as *viable*. We associate the term viable with whether a contour observes the rules imposed or not. By observing these rules we could safely say that all viable contours do sound melodic and flow smoothly. This observation can be intuitively demonstrated by listening to the results of the algorithm. The issue of whether the contours actually sound beautiful, or whether they express some kind of emotion is another issue altogether. Whilst it is true that certain compositional techniques can give melodies some emotional character[4], in this paper we have not considered them and left the issue as a potential next topic of research. We conclude this work by suggesting some techniques and future projects than can augment the simple pitch contours we generated here and use them as the basis of fully fledged musical compositions:

- Apply note grouping techniques and rhythmic unit presets to the pitch contour for it to become a complete rhythmic melody line.

- Using species counterpoint techniques to enrich the melody harmonically. Chord progression rules may be derived by studying common progressions used in the CPP.

- Observing cadences when harmonising the melody.

- Observing orchestration guidelines when determining the relationships between voices.

- Using instrumentation principles when choosing instruments to play a given voice. For example certain instruments possess timbres that lend themselves to a more dramatic score.

[4]For example, it is a known 'fact' that composing in a major key usually results in 'happy sounding' scores. Composing in minor keys is usually associated with music of the more 'sad' kind.

5. REFERENCES

[1] Anthony Ashton. Harmonograph: A visual guide to the mathematics of music. Wooden Books.

[2] Benjamin Piekut. From No Common Practice: The New Common Practice and its Historical Antecedents. American Music Center. http://www.newmusicbox.org/page.nmbx?id=58tp01

[3] The Associated Boards of the Royal Schools of Music. Rudiments and Theory of Music.

[4] Michael Miller. The Complete Idiot's Guide to Music Theory. Alpha Publishing.

[5] William Lovelock. First Year Harmony. Hammond Textbooks.

[6] Charles Dodge, Thomas A. Jerse. Computer Music: Synthesis, Composition and Performance. Second Edition. Schrimer.

[7] David Cope. Virtual Music: Computer Synthesis of Musical Style. The MIT Press.

[8] John Chuang. Mozart's Musikalisches Würfelspiel. http://sunsite.univie.ac.at/Mozart/dice/.

[9] David E. Goldberg. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley Professional.