Music Composition Using Genetic Evolutionary Algorithms

M. Marques, V. Oliveira, S. Vieira, AC Rosa
Laseeb-ISR-IST
Av Rovisco Pais 1, Torre Norte 6.21
1049-001 Lisboa, Portugal
Email: acrosa@isr.ist.utl.pt

Abstract - Genetic and evolutionary algorithms are a possible way of searching for the solution to problems in large dimension search spaces. In this work the authors have applied these methods to the generation of musical sequences using melodic and musical theory concepts such as fitness function.

1 Introduction

The goal of this work is to produce musical compositions of enough quality so that they can be appreciated in a similar way to other musical pieces, thus allowing the automatic creation of music that can be used in many situations such as ambience music or incidental parts. Unlike other works, the method used can be applied without the need of any external user intervention, the providing of starting sequences or melodies, or Subjective fitness feedback, which effectively means creating something from scratch.

Modern musical composers follow many times some simple rules during their creative processes. These usually refer to the intervals between notes, to the notes that should be used in each tone and in each musical style and to the rhythmic articulation types used.

By studying these rules one may be led to the creation of various algorithmic composition methods used as a fitness function. The outcome of the process is a small set of sample compositions with best fitness. These samples are played and/or presented in musical notation to the user in order to be appraised. The user makes the decision to retain these samples or not. The user also has some control of the creative process through the parameters supplied to the fitness function.

Evolutionary Algorithms, have been shown to be an effective method to find solutions in wide search spaces, therefore they are quite suitable to be applied to this kind of problems, being only necessary to use an appropriate fitness function.

2 State of the Art

Western music is extremely dependent on the mathematical relationships between sounds. As such, several efforts have been made to apply algorithmic techniques to the creation of musical works. A good summary on the currently known methods for algorithmic composition can be found on (Beckert 1997). This document covers some of the basics of GA composition and also focuses on some of its problems. The type of implementation described makes use of an initial musical sequence. The GA then evolves a sequence of operations that transform this initial sequence into a new one. Similar techniques can be found in (Alpern 1995), where a musical piece is evolved which responds to simple musical phrases with different ones, in a "call-response" manner. See also (Goldberg 1991) for a similar application on creating a bridge to connect supplied initial and final patterns.

Other works can be found that do not depend on an initial sequence. For example, (Fujinaga 1994) uses a technique known as granular synthesis. A musical piece is build up as a sequence of short sound waves (called "grains"). The GA chromosome is composed of a sequence of grain parameters, which evolve until a more "musical" sequence appears. (Johanson 1997) proposes a different approach, using genetic programming. This method allows structured sequences to appear, adding repetitions, transposed sequences and other elements that are often heard on compositions by humans.

3 The Algorithm

Many methods derived from the Standard GA (SGA) have been used. Further insight on the GA can be found on (Goldberg 1989). All these methods have in common the usage of a large space of possible solutions to a given problem, with the search being made by variation, starting from initial points in that space and progressing until higher values of the fitness function are found.

After trying the Standard GA with not very promising results, both regarding the number of evaluations required and the best fitness values attained, the authors decided to use in this work a variant known as "Familial Competition", which proceed to supply better results.

This algorithm starts with a random point population in the search space, encoded in an appropriate way for the application of the algorithm, in what is called the genome. Points are then randomly grouped in pairs. Each of these pairs is then subject to a crossover process, where the two partners exchange part of the genome (crossover). After that, they suffer a mutation operator where some of their genes may be replaced by random values, resulting in new individuals. These new individuals, the children, are then evaluated and compared to their parents, with the two best (among all four) being selected to pass on to the next generation. The process is then repeated until an individual is found whose fitness is higher than a predefined value or until a certain number of iterations are reached.

3.1 The Genome

A genome was used to represent each individual that is an almost direct representation of the corresponding musical part. The musical piece is divided into several voices, each corresponding to a different melodic line, like, for example, the different instruments in an orchestra. Each voice is then divided into fixed duration time units, resulting in a two-dimensional representation of the piece. Each gene fills the space belonging to a time unit and indicates what type of sound is heard in that interval. As a way to encode the different duration of notes and pauses, the following scheme was used:

- Notes were encoded according to (Oliveira 1998), assigning to each sound a whole number. Whole numbers between –48 and 49 were used, corresponding to 8 octaves, enough for the human hearing range. Numbers between 0 and 11 corresponded to the octave beginning in middle C. Transposition to the other octaves is made by adding 12×(octave number).
- Pauses were encoded as numbers between –128 and –100 and between 100 and 127.
- Notes or pauses with duration superior to a time unit are "followed" by a symbol meaning that their sound spans across the unit of time that this symbol occupies. This symbol may be repeated until the note's duration is complete. This "hold" symbol is encoded as a number between –99 and –49 and between 48 and 99.

This way, all numbers between –128 and 127, which may be represented as a single byte, are used in the encoding. The next figure summarises this representation.

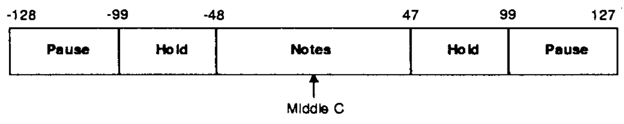


Figure 1. Genome representation

This way, a high probability is assured for "pause" and "hold" symbols, which could be less frequent than desired, due to a smaller value given to them by the fitness function, which is described later on.

3.2 Operators

The crossing and mutation processes referred to earlier, constitute the algorithm's variation operators.

Due to the specific nature of the present problem, multipoint crossover and mutation operators were implemented as well as 2 variants. They will be described in the following paragraphs.

3.2.1 Multipoint Crossover

This type of crossover is achieved in the following way:

The genes to be crossed are randomly chosen. The number of crossing points can be changed in the program, as well as the correspondent probability and priority (these two parameters will be discussed later on).

After the choice, the exchange of genetic material is made, as each piece exchanges notes, pauses and "hold" symbols with the other. This is shown in figure 2.

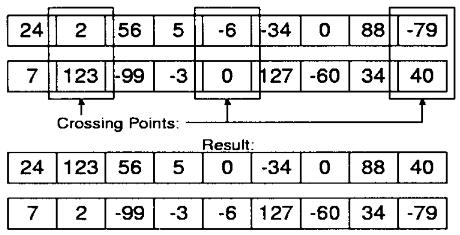


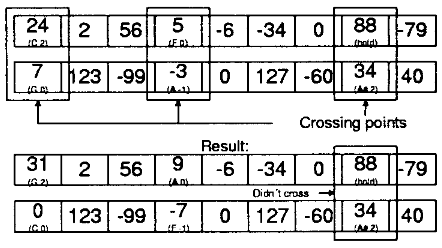
Figure 2. Multipoint Crossover

3.2.2 Note Crossover

This crossover is also a multipoint one but the exchange of information is made in the following way:

- If one of the genes to be crossed is not a sound then crossing doesn't take place at that point.
- The octave of each element to be crossed is kept because only the information regarding the notes is exchanged.

Figure 3 shows how the note crossing is made. It should be noted that the third crossing would not happen because of the reasons previously described.



24 (C 2) | 2 | 56 | 5 (F 0) | -6 | -34 | 0 | 88 (hold) | -79
7 (G 0) | 123 | -99 | -3 (A -1) | 0 | 127 | -60 | 34 (A# 2) | 40
Crossing points
Result:
31 (G 2) | 2 | 56 | 9 (A 0) | -6 | -34 | 0 | 88 (hold) | -79
Didn't cross
0 (C 0) | 123 | -99 | -7 (F -1) | 0 | 127 | -60 | 34 (A# 2) | 40

Figure 3. Note Crossover

3.2.3 Octave Crossover

This operator is similar to the Note Crossover operator, but in this case the exchange is made in each gene's octave, keeping the original note. Again the gene is required to be a sound for the crossing to take place. Figure 4 shows this type of crossing.

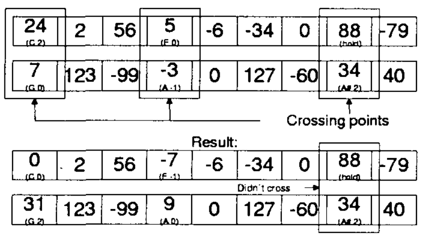


Figure 4. Octave Crossover

3.2.4 Multipoint Mutation

In the mutation operator the genes to be mutated are randomly chosen, and are then substituted by a random number between –128 and 127. The probability and priority of this operator can be changed in the program, as in the other variation operators.

3.2.5 Note Mutation

This operator is similar to the previous, but in genes to be mutated only the note is changed keeping the octave.

3.2.6 Octave Mutation

In this operator the mutated gene will keep its original note, changing its octave.

In the following figure the three mutation operators are shown:

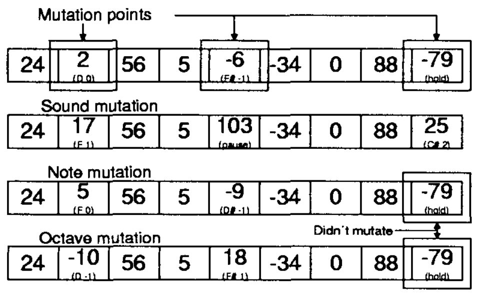


Figure 5. Mutation Operators

Note and octave crossover and mutation give the algorithm an ability to perform a local search, since they correspond to sensible variations in the individual's characteristics.

3.3 Operators – Parameters

All six operators described have in common a series of parameters, independent for each operator and changeable in the program.

3.3.1 Crossover and Mutation Probabilities

These are the usual probabilities, that is, if for example a pair of chromosomes is chosen for note crossover, which will only take place if a random test ("unbalanced coin") is favourable.

3.3.2 Crossover and Mutation Priorities

In order to cross or mutate individuals when there is more than one process to do so, the same individuals shouldn't be crossed or mutated more than one time in the same generation. Thus a priority scheme was implemented, so that only one kind of operator is applied in each generation. Each operator's priority value varies between 0 and 1, but the sum of the priorities must always be 1. This works like a kind of roulette that chooses which operator should be applied in the present generation, to the chromosome(s).

3.3.3 Percentage of genes to Cross/Mutate

This option allows us to choose how many genes should be crossed or mutated in an individual.

There's also an option to directly disable each of the operators.

4 Fitness

The fitness function used was built from some simple rules that have been used many times by composers.

Evaluation is made in three main parts:

- Harmony, that values intervals between notes played at the same time by different instruments (voices);
- Tone, which values how suitable a note is for the chosen tone and scale.
- Melody, that values the intervals between consecutive notes.

To evaluate harmony, each note is grouped with the corresponding notes from the two following voices, so that a 3 note chord can be formed. This process is shown in figure 6.

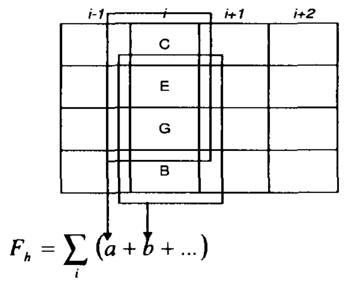


Figure 6. Fitness calculation

The measure of the intervals between that note and each of the two other's (given by the difference between the numbers that represent the notes, according to (Oliveira 1998)) is used to search for a fitness value in a table. In the present experiments a table was used that gave maximum merit to the different inversions of major and minor chords (the most common chords in modern music) and zero to other possible combinations. Other tables result in different sonorities. Figure 7 shows one of the tables used in the program.

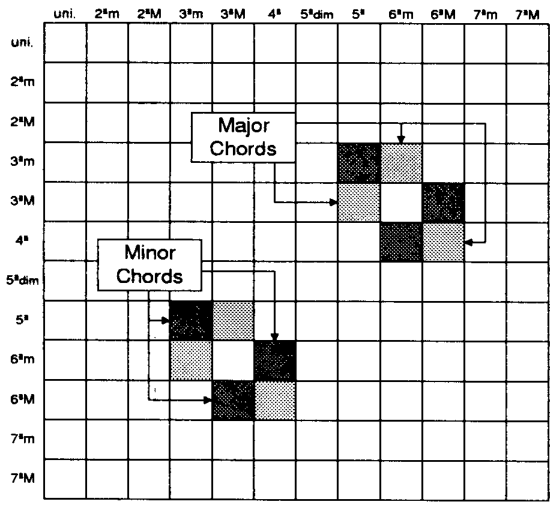


Figure 7. Table of fitness values

Tone was evaluated simply by giving maximum value when a note belonged to the chosen scale and tone, and zero to all other cases (some scales have ambiguous notes punctuated with middle values). Figure 8 shows two of the scales used, where one of them has values different from 0 and 1.

C Major Diatonic Scale (uniform weights)											
C	C#	D	D#	E	F	F#	G	G#	A	A#	B
1.0	0.0	1.0	0.0	1.0	1.0	0.0	1.0	0.0	1.0	0.0	1.0
0	1	2	3	4	5	6	7	8	9	10	11

C Blues Scale (uniform weights) with major third (half weght)											
C	C#	D	D#	E	F	F#	G	G#	A	A#	B
1.0	0.0	0.0	1.0	0.5	1.0	1.0	1.0	0.0	0.0	1.0	0.0
0	1	2	3	4	5	6	7	8	9	10	11

Figure 8. Tone scale

Melody was measured in a similar way to tone, but this time evaluating the interval between two consecutive notes. Figure 9 shows the evaluation for the diatonic scale. All the scales and tables presented here (and others that were not mentioned) are available in the program, so that one can change the musical style.

Intervals in the Major Diatonic Scale

uni.	2ªm	2ªM	3ªm	3ªM	4ª	5ªdim	5ª	6ªm	6ªM	7ªm	7ªM
1.0	0.28	0.7	0.42	0.56	0.84	0.14	0.84	0.56	0.42	0.7	0.28
0	1	2	3	4	5	6	7	8	9	10	11

Figure 9. Diatonic scale

The value of the fitness function is given by the weighted sum of each one of the three components to all of these notes, divided by the number of notes and voices.

5 Results and Conclusions

5.1 Population size

The algorithm converges to the highest fitness values with populations of at least 600 individuals, as can be seen in figure 10, where the values shown are the averages of the maximum fitness values from ten runs.

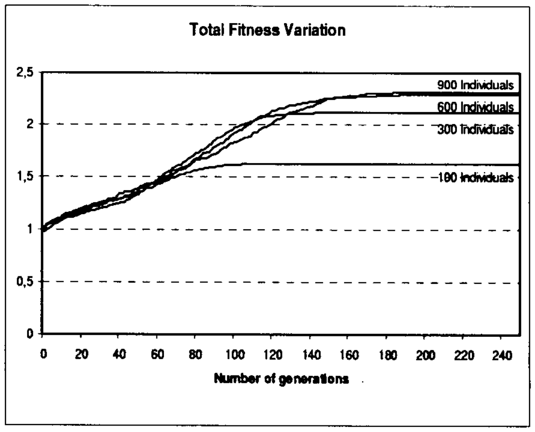


Figure 10. Convergence results

Figure 11 shows how the three components of the fitness function vary. Tone evaluation usually has the fastest convergence, reaching about 95% of the maximum value in less than 100 iterations. Both tone and melody reach very high values, even with smaller populations. Harmony, on the other hand, requires more than 600 individuals to attain good results.

5.2 Variation operators

Each curve in this section was attained at an averaging of ten runs, with a population of 600 individuals.

From figure 12 it can be seen how the different types of operators used affect convergence. Each curve results from an average of ten runs using only a particular kind of operator. The best values are obtained with the standard operators, with note operators being, by far, the worst option.

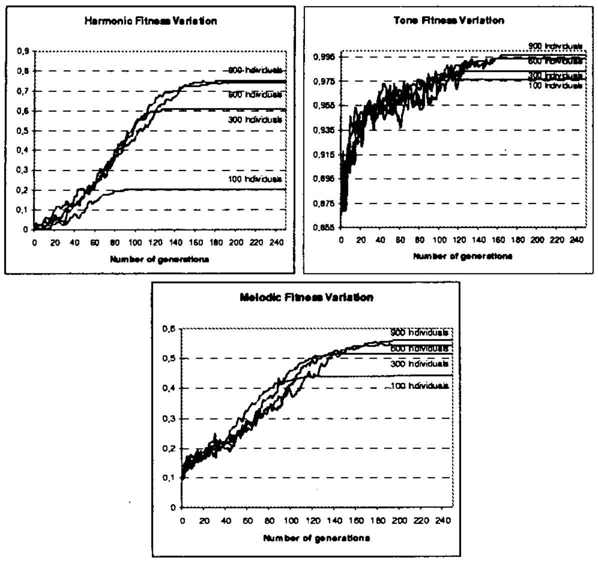


Figure 11. How do the fitness components vary

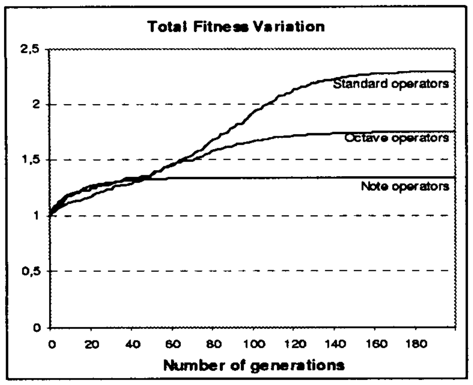


Figure 12. Total fitness variation

Looking at the fitness components in figure 12 offers some insight as to why this happens. Note operators give the best results in tone evaluation, closely followed by standard operators. This is because the octave component is not important for the method used to evaluate tone. On the other hand, melody evaluates the best when octave operators are used. Melodies that jump from one octave to another are usually unpleasant, and the melody fitness function reflects this, giving lower values for consecutive notes that spread across octaves. Standard operators give the best results for harmony evaluation.

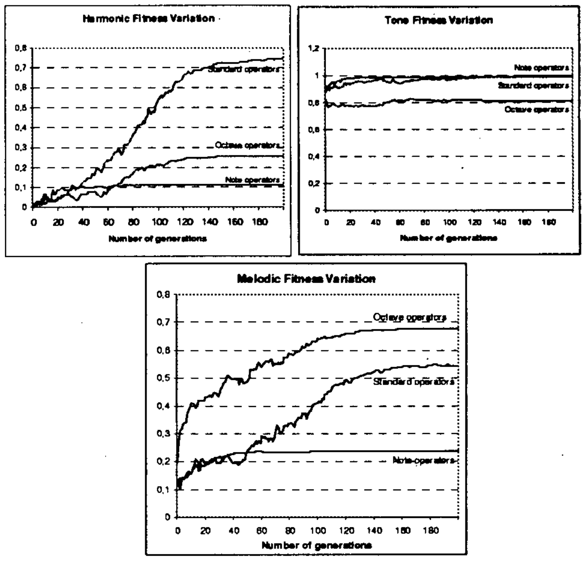


Figure 13. Fitness variation

This analysis led us to conclude that octave operators are required for good melodic convergence, but those standard operators are enough for good results in both tone and harmony.

The results in figure 14 are a comparison of what can be attained using standard operators alone and together with octave operators. The results obtained reach similar values of the fitness function, but with very different values for each of the harmonic and melodic components, as could have already been predicted from the previous results.

This way, a more melodic tune can be found by increasing the priority for the octave operators, while a piece where harmony prevails can be accomplished by decreasing it.

The best sequence from each of the runs that were used in these experiments was almost always pleasant to the listeners, although only experiments with short sequences were made.

The authors believe to have shown that it is possible to evolve pleasant musical sequences using only a small set of simple rules.

Figure 15 shows an example with 3 voices and 20 time units, of the same kind that had been used in the previous experiments, written in music notation.

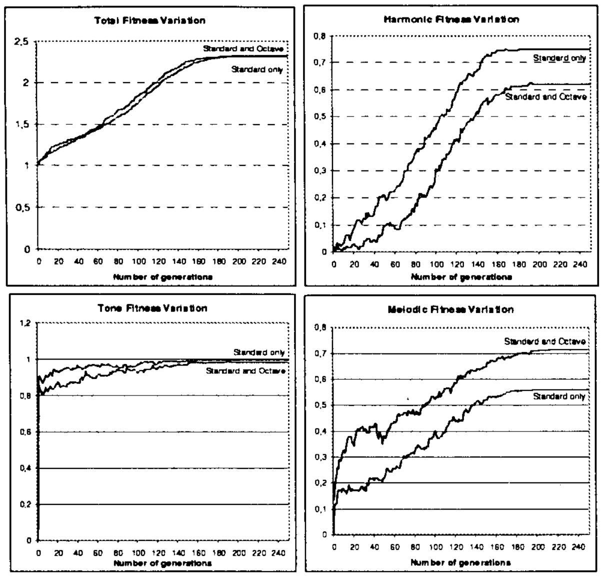


Figure 14. Comparison of results using standard operators alone and together with octave operators

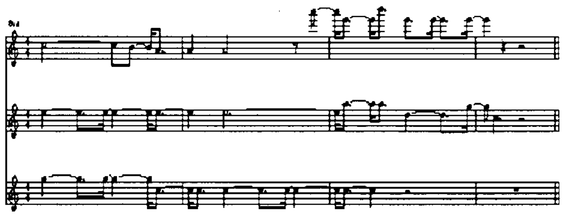


Figure 15. Example solution

More of the examples used can be found, in MIDI file format, in the Internet on (Rosa 1999).

References

Alpern, A. (1995) "Techniques for Algorithmic Composition of Music", Hampshire College Divisional Examination, Humanities and Arts.

Beckert, D. (1997) "Algorithmic composition", BSc thesis, University of Reykjavik.

Fujinaga, I. and Vantomme, J. (1994) "Genetic algorithms as a method for granular synthesis regulation", Proceedings of the International Computer Music Conference. pp 138-141.

Goldberg, D. (1989) "Genetic Algorithms – in Search, Optimization and Machine Learning", Addison-Wesley Publishing Company.

Goldberg, D. and Horner, A. (1991) "Genetic Algorithms and Computer-Assisted Music Composition", Proceedings of the Fourth International Conference on Genetic Algorithms, pp 437-441.

Johanson, B. E. (1997) "The GP-Music System: Interactive Genetic Programming for Music Composition", University of Birmingham, Second-Semester Mini-Project Report.

Oliveira, J. (1998) "Analytical Theory of XXth Century Music".

Rosa, A. C. (1999) "Sample MIDI files", http://www.geocities.com/ResearchTriangle/Station/1232/geamusic.html