2C4-IOS-3c-4

A Comparison between Genetic and Memetic Algorithm for Automated Music Composition System

Mondheera Pituxcoosuvarn, Roberto Legaspi, Rafael Cabredo, Ken-ichi Fukui, Koichi Moriyama, Noriko Otani*, Satoshi Kurihara, Masayuki Numao

The Institute of Scientific and Industrial Research, Osaka University
Faculty of Informatics, Tokyo City University*

Automatic music composition has been a challenging, interesting, and yet still much to be explored task primarily because it is hard to distinguish which song is good or bad which significantly impedes the automated composition process. Despite this difficulty, automated music composition would benefit many groups of people who ought to use a piece of their own music, as composed for them by an AI system with compositional intelligence, without someone else's copyright for some purpose such as a music piece for a commercial, or a song played in the background of a presentation. Our composition system composes eight-bar tracks, based on western music theory and listener evaluation. We present here the use of memetic algorithm, comparing to using the conventional genetic algorithm. The same representation and evaluation for both techniques are used because of the similarity of these two algorithms. The main difference of memetic algorithm with genetic algorithm is the local search process. Both algorithms are implemented separately to spot the difference between the results then we evaluated the algorithms. When the outcomes are compared, we found that the use of memetic algorithm performs better in terms of quality of musical piece and convergence speed.

1. Introduction

Music composition is normally done by music experts and music composers. They use both feeling and musical knowledge to come up with good pieces of music. In some cases, we might want to get a generated music without the need of hiring a composer, and achieving this kind of intelligence has become one of the computer scientists' dreams. Automatic music composition would benefit many groups of people who ought to use a piece of their own music composed for them by an AI system with compositional intelligence such that it is not under someone else's copyright, for a purpose such as for a commercial or to be played in the background in a presentation.

Automated music composition problem is challenging and interesting for computer scientists. Unlike neither scientific nor mathematical problems, the beauty of music is difficult to judge since it is art and subjective. There is no way that many people will judge a song in the same way. Everything is about how an individual feels. These are the reason why it challenges us to work on this project. We have tried to automate the composition of a completely new song from scratch, but then the generated music should not merely consist of a set of random notes—it has to incorporate a certain degree of creativity based on the music theory. In real life, a person who wants to try writing a piece of musical melody either has to understand music theory or to be really talented.

Genetic algorithm (GA) and memetic algorithm (MA) were chosen in order to solve this problem and to be compared against each other. MA includes a local search process, which might be helpful to solve problems that we already know how to improve. To illustrate, we basically know how to improvise a song by applying music theory. Then we create a set of rules extracted from music theory that will become our MA local search function. Without any idea of how to improve an individual, the local search might not be useful.

Contact: Mondheera Pituxcoosuvarn, The Institute of Scientific and Industrial Research, Osaka University, 8-1 Mihogaoka, Ibaraki, Osaka, 567-0047, Japan, Tel: +81-6-6879-8426, Fax: +81-6-6879-8428, mondheera@ai.sanken.osaka-u.ac.jp

2. Background

2.1 Music Background

This section outlines fundamental western music knowledge that is needed when automating composition using artificial intelligence techniques. To compose a piece of music, we need to arrange symbols including notes and rests. In this research project, only western music theory is used for international standardization.

2.1.1 Note and Rest

A note is a fundamental and important part of music. It has two main characteristics: pitch and duration. Music pitch often refers to the frequency of the sound. In science, pitch means higher frequency and lower frequency. Note value or note duration is a value showing how long a note is meant to be played. Notes are normally written on "staffs" or "staves", each of which consists of a set of five horizontal lines. In western music theory, songs are written based on twelve notes including C, C#/Db, D, D#/Eb, E, F, F#/Gb, G, G#/Ab, A, A#/Bb, and B. In many countries especially in Latin America, Eastern Europe, and Southern Europe, another system of naming notes is used, which are Do for C, Re for D, Mi for E, Fa for F, Sol for G, La for A, and Ti or Si for B. There are 5 notes that have two names that are represented as black keys on a keyboard.

Apart from the pitch, another important part of a note is the note value and note duration. Note duration, as shown in the figure 1, is about how long the note will be played, represented by the shape of the note symbol. One whole note has a duration equals to two half notes. Half note duration is equal to two-quarter notes. An eights notes duration equals two sixteenth notes. There are more types of note durations but these five are the most used ones.

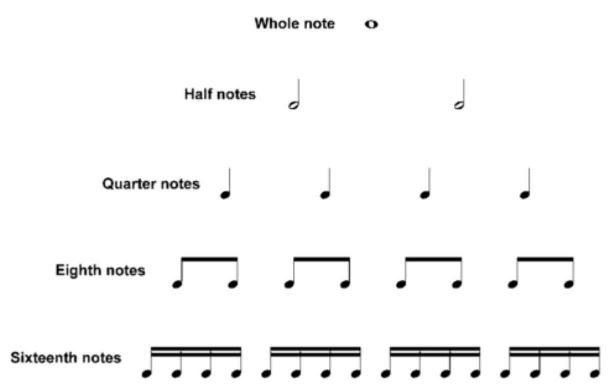
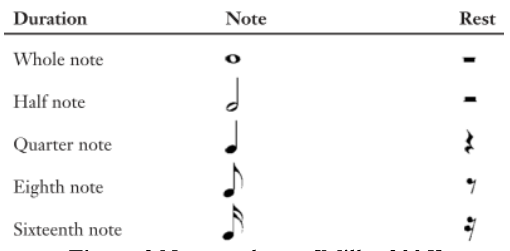


Figure 1 Note duration chart [Miller 2005]

Corresponding to the notes, there is another set of symbols, leaving some part of the song silent, called "rest". Those rests are displayed below. However, the exact duration in terms of actual time of the notes and the rests is defined by the tempo.



Duration	Note	Rest
Whole note	𝅝	𝄻
Half note	𝅗𝅥	𝄼
Quarter note	♩	𝄽
Eighth note	♪	𝄾
Sixteenth note	𝅘𝅥𝅯	𝄿

Figure 2 Notes and rests [Miller 2005]

2.1.2 Interval

Interval is the distance between 2 notes. The smallest interval for Western music is a semitone or a half step. There are a number of types of interval depends on the size and the quality. Interval size is about the space between those 2 notes. For example, there are 5 notes from C to G, so this interval is "fifth" or "5th Interval". The other name and size is shown in the **Figure 3.**

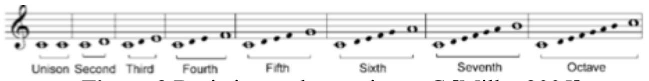


Figure 3 Basic intervals, starting at C [Miller 2005]

Regarding the quality of an interval, there are various types depending on how well those two notes go together. The biggest categories are "perfect interval" and "non-perfect interval". The perfect intervals are unison, fourth, fifth, and octave. When a perfect interval is lowered by a semitone, it comes to be "diminished". But if it is raised by a semitone, it is called "augmented". For non-perfect intervals, there are second, third, sixth, and seventh which can be either major or minor. Major intervals are the natural state in major scales. If a major interval is raised by a semitone, it becomes "augmented", but if it is lowered by a semitone, it changes into minor. On the contrary, if a minor is raised by a semitone it turns into "major" but if it is lowered by a semitone it becomes "diminished".

2.1.3 Scale

A scale is composed arbitrary of a series of notes. Nowadays mainly two scales that are still used to compose a song generally: major and minor scales. Every scale starts from a note and end at the same note in one-higher octave. There are twelve notes in one octave but there are only eight notes in a scale. To construct a scale, specific intervals are used. In use, composers normally use note in a specific scale to write a piece of music.

In a major scale, C Major contains C, D, E, F, G, A, B, C. The first note in the scale is called "Tonic" and the intervals between each notes are 2, 2, 1, 2, 2, 2, 1 semitone respectively.

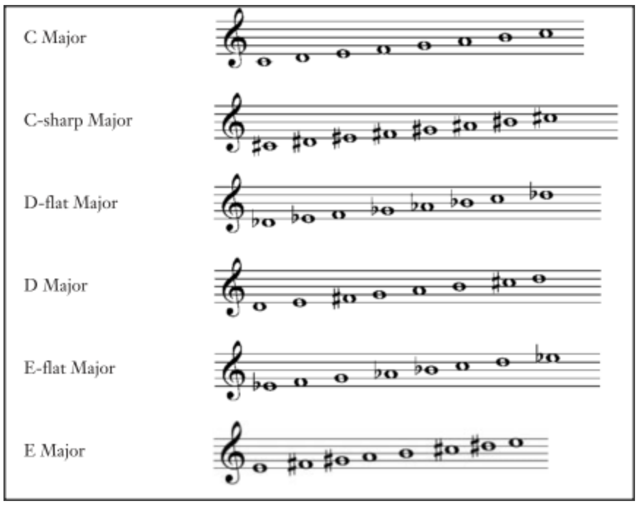


Figure 4 An Example of major scales [Miller 2005]

Minor mode is more complicated than major mode. There are three types of minor scale: "natural minor", "harmonic minor", and "melodic minor". As with the major scale, interval plays the only role here. A natural scale is composed of a tonic and the note with 2, 1, 2, 2, 1, 2, 2 semitones in order. The harmonic minor scale is almost the same as the natural minor. The small difference is the seventh note of this scale is raised by a semitone, which makes the interval between the sixth and the seventh notes become three semitones. It sometimes sounds abnormal and is difficult to sing. As a consequence, the melodic minor scale adjusted the too big interval, so the intervals turn into 2, 1, 2, 2, 2, 2, 1 semitones.

2.1.4 Motif

Motif is found in many parts of the composition. It is a very short sentence of rhythm or melody and it is repeated many times. Pitches can be adjusted but it will keep the rhythm as its characteristic. A great example of using motif is Beethoven's Symphony No.5.

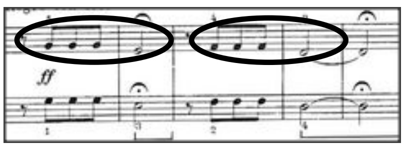


Figure 5 The motif from Beethoven's Symphony No.5

2.2 Related Works

Recently, there have been several method presented for both automatic music composition and automatic improvisation. GA and other hybrid methods of GA are used for music composition by many researchers. [Marques 2000] [Unehara 2001] [Sheikholharam 2008] [Maeda 2010]

By the way, other techniques are able to solve this problem. Several researches are done using various methods, especially AI algorithms. One of those is Chiu[2006]'s work that presented a composition approach by discover the patterns and rules in existing songs then automatically create a new piece which is similar to the original songs.

MA is a fairly new population based algorithm and there exists not so many works using this algorithm. This algorithm could be seen as a hybrid or developed method of GA. The term "meme" was coined by Dawkins[1976], an author, ethologist and evolutionary biologist. This term refers to a unit of cultural transmission similarly to gene. As a result, "memetics" is corresponding to "genetics". The first person that applied meme concept to problem solving is Moscato[1989]. His algorithm is named memetic algorithm (MA). Memetic algorithm was applied to automated music composition without human evaluation. [Wells 2011] Their work included a few rules from music theory, used for fitness calculation and local search.

3. Automated Music Composition System

This project applies GA and MA as main method for music composition. There is evaluation by human included in order to estimate the quality of the composed piece since evaluation using only few rules from music theory is not enough to judge the results. Moreover music theory is just a concept we cannot make sure that if we follow all the theory we will get a good song. For these reason, human is needed to evaluate the songs

3.1 Composition Boundary

Music system used in this work is based on western music theory with twelve notes, A to G#. To reduce the complexity, only major scale is employed in the composition. The shortest duration of note is sixteenth and the longest duration of note is the whole. The same as most simple songs, time signature 4/4 is applied and the length of the composition is set to eight bars. The output of the composition is music melody in midi file format.

3.2 Genetic Representation

Our representation is based on GA. In one generation there are a number of chromosome or population. Each chromosome, representing one song, contains a number of notes. Each note has two main characteristics: pitch and rhythm value, as displayed in the **Figure 6**.

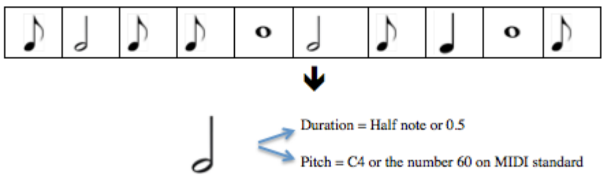

Duration = Half note or 0.5
Pitch = C4 or the number 60 on MIDI standard

Figure 6 Genetic representation

3.3 Genetic and Memetic Algorithm

Since the algorithms used in this experiment, GA and MA, have very similar processes, all the identical processes will use the same functions and parameters. MA is roughly a concept of hybrid GA with individual improvement. The pseudo-code* below shows both algorithms. These algorithms are adjusted for this automated music composition problem.

GA Pseudo-code:	**MA Pseudo-code:**

Population P = initialize();
while(!stopping_conditions)
  {
  offspring O = crossover(P);
  mutate(O);
  evaluate(P,O)
  P = regenerateP (P,O);
  }
	Population P = initialize();
	while(!stopping_conditions)
	  {
	  local_search(P);
	  offspring O = crossover(P);
	  mutate(O);
	  evaluate(P,O)
	  P = regenerate (P,O);
	  }

The system first creates P population randomly then starts our development processes until the stopping criterion is met. Iteration counter is the only one parameter for our stopping condition. Only MA includes individual improvement process, called local search. In this specific problem, this function does not really do local search but try to improve individuals. The common processes of both GA and MA are crossover, mutation, and generation reproduction. Simple crossover is done to create offspring O. The system will firstly pair parents randomly and choose one point to fix only one point for cross over operation. Crossover rate for this experiment is 100 percent. Mutation is done by choosing mutate points randomly on each offspring O based on the mutation rate. All offspring are mutated. Each child will be adjusted one note pitch and one note duration once for mutation randomly. After crossover and mutation process both population P and offspring O fitness will be calculated then each the offspring will be compared with its parents. If an offspring gets the better fitness value the offspring will be chosen for the next generation. Unless a parent has more fitness value, the parent will continue going on the next generation. Before regeneration, each individual has to be evaluated, then chosen to survive on the new generation.

3.4 Fitness Function

Songs from both GA and MA is evaluated by a fitness function. This function is from extracting major characteristic of a good music piece from music theory and also from the evaluation of the listener. Fitness function of this system is described as the equation below

$$\text{Fitness} = \{(W_1\times\text{InScaleScore})+(W_2\times\text{LengthScore})+(W3\times\text{IntervalScore})+(W_4\times\text{TonicScore})+(W_5\times\text{ListenerScore})\}$$

Fitness value is calculated from five factors and every score is multiplied by its weight (W_1-W_5). Each score maxima is 1. InScaleScore is computed from the ratio of the number of in scale notes and the number of all notes on the chromosome.
LengthScore is the value of the duration the exceeds 32 beats or be less than 32 beats divided by 4, since 32 is equal to eight bars in 4/4 and 4 is the largest possible difference. IntervalScore is

calculated from the amount of intervals that less than 5th divided by the number of all interval found in the music piece.
TonicScore only checks the first and the last note whether they match the tonic or not. Each tonic note costs half the score, which is totally 1 from both notes. ListenerScore is ranged from 1 to 5 based on the evaluator. The listener could be a general listener or a music expert who grades the song. The raw score from listenner is divided by 5 to be easier to calculate the total fitness.

3.5 Local Search

There are five functions to improve individually. Each function is decided to be used or not depends on randomly choosing based on the probability parameters except the last function adjustNoteToSequence(). We also apply probability base random but in different way.

Pseudo-code for local search:

```
public void doLocalSearch()
{
 if(getABoolean(ProbInScale))   {adjustNoteToScale();}
 if(getABoolean(ProbMotif))     {createMotif();}
 if(getABoolean(ProbStartEnd))  {setStartEnd();}
 if(getABoolean(ProbDuration))  {adjustDuration();}
 adjustNoteToSequence();
 }
```

getABoolean(Double) function coins a Boolean value randomly, depending on a double-type input. This value controls the weight of the probability for calling function**.** Once adjustNoteToScale() function is called, all notes in a chromosome will correct out of scale note to the scale. For example to compose a C Major song, the note C# does not belong to C Major scale. C# will be raised to be D or lowered to be C depending on the random Boolean value. **While** createMotif() function randomly chooses one bar in the chromosome except the last bar. And copy all notes in that chosen bar and paste them next to the chosen bar. There are two main methods of creating music motif that are also chosen randomly. First, only copy and paste without changing any value. Second, shift all notes in the copied bar up or down. setStartEnd() is a function that set the first and the last note of a chromosome to be the tonic of the key. For instance, the note C is the tonic of both C major and C minor scale. adjustDuration() function changes the too long or too short chromosome duration. For the song shorter than eight bars, the program will search for one smallest duration note from left to right and adjust its duration value by two times. For the music that longer than eight bars, the program will search for the longest note instead and lessen the duration by half. This function will adjust only one note when it is called. In every iteration, the function named adjustNoteToSequence() is used, unlike the previous functions. In the function it will look at each note from left to right and it may adjust that specific note or just leave the note the same depends on the weighted random value. If a note has to be changed because of its random value, it will be set either closer to the previous note or closer to the next note. The new note pitch will be not further than 3 pitches compare to one of its neighbors.

4. Result

After running the algorithms for 15 iterations, we got a number of developed music. **Figure 7** shows a part of the result from running GA and **Figure 8** shows a section of music composited from MA.

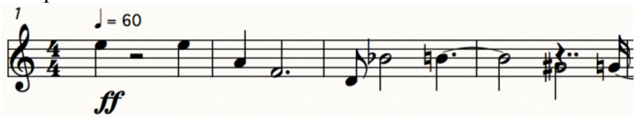


Figure 7 A part of the result music piece using GA

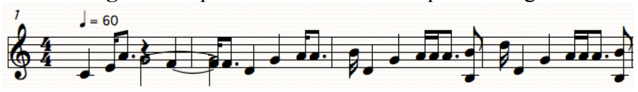


Figure 8 A part of the result music piece from MA

To compare the performance of each algorithm, score from user evaluation and the generated music scores of the first and the last generation is shown in the table below.

Composited Music	Genetic Algorithm		Memetic Algorithm	
	Average Score	Best Score	Average Score	Best Score
Random	1	1	1	1
Result	1.2	2	1.8	3

For MA, the listener mainly scored 1 for the first generation but the 10th generation music got about 2-3 point out of 5. MA shows that it is also an effective method to develop music composition system, compared to GA.

5. Conclusion

After this experiment, we found that MA might fit automated music composition problem more than GA since it performs better and have the better score form the listener. The local search is the characteristic of MA, compared to GA and this is useful for the problem that we know a little bit how to improve it individually. In music composition, we already derived music theory, which has been accepted worldwide. We know that when we apply music theory to the composition we will get the better result and it is not difficult to create rules from the theory. Conversely, we cannot totally believe the music theory since it is written very long time ago and has not been developed lately. Listener evaluation can help us finding the fault of the theory if the fitness score created from the music theory is far different from the listener score. Moreover, the convergence of MA is quite fast. We can get the better song within a few iterations that could reflect that local search might help the algorithm converge faster

5.1 Future Work

The local search function and the fitness function still can be further develop because there are many rules and theory of music that can help us improve this automated composition.

To make the evaluation easier for the listener, listener evaluation model could be created to help the listener not to listen to every song in the every generation, which is time consuming and might bore the listener and could affect our listener score.

Acknowledgments

This work is also advised by Dr. Boomserm Kaewkamnerdpong and Assoc. Prof. Tiranee Achalakul. King Mongkut University of Technology, Thailand.

References

[Chiu 2006] Shih-Chuan Chiu, Man-Kwan Shan, Computer Music Composition Based on Discovered Music Patterns, In Proceeding of IEEE International Conference on Systems, Man and Cybernetics, SMC '06, vol.5, no., pp.4401,4406, 8-11 Oct. 2 006, Taipei, IEEE, 2006

[Dawkins 1976] Richard Dawkins, The Selfish Gene, Oxford Press, 1976

[Maeda 2010] Maeda Y., Kajihara Y., Rhythm generation method for automatic musical composition using genetic algorithm, In Proceedings of IEEE International Conference on Fuzzy Systems (FUZZ), 2010, vol., no., pp.1,7, 18-23 July 2010, IEEE, 2010

[Marques 2000] Marques M., Oliveira V., Vieira S., Rosa A.C., Music composition using genetic evolutionary algorithms. In Proceedings of the 2000 Congress on Evolutionary Computation, 2000, vol.1, no., pp.714,719 vol.1, IEEE, 2000

[Miller 2005] Michael Miller, The Complete Idiot's Guide to Music Theory, USA, Alpha Press, 2005

[Moscato 1989] Pablo Moscato, On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms, Tech. Rep. Caltech Concurrent Computation Program Report. 826, USA, California Institute of Technology, Pasadena, 1989

[Sheikholharam 2008] Sheikholharam P., Teshnehlab M., Music Composition Using Combination of Genetic Algorithms and Recurrent Neural Networks, Eighth International Conference on Hybrid Intelligent Systems, 2008. HIS '08, vol., no., pp.350,355, 10-12 Sept. 2008, IEEE, 2008

[Unehara 2001] Unehara M., Onisawa T., Composition of music using human evaluation, In Proceedings of The 10th IEEE International Conference on Fuzzy Systems, 2001, vol.3, pp.1203, 1206, IEEE, 2001

[Wells 2011] Derek Wells, Hala ElAarag, A novel approach for automated music composition using memetic algorithm in ACM SE 11, Mar 24-16 2011, GA, USA, ACM, 2011