Music Recombination using a Genetic Algorithm

Sanjay Majumder
Indiana University-Purdue University Indianapolis
sanjay@iupui.edu

Benjamin D. Smith
Indiana University-Purdue University Indianapolis
bds6@iupui.edu

ABSTRACT

This paper presents a new system, based on genetic algorithms, to compose music pieces automatically based on analysis of the exemplar MIDI files. The aim of this project is to create a new music piece which is based on the information in the source pieces. This system extracts musical features from two MIDI files and automatically generates a new music piece using a genetic algorithm. The user specifies the length of the piece to create, and the weighting of musical features from each of the MIDI files to guide the generation. This system will provide the composer a new music piece based on two selected music pieces.

1. INTRODUCTION

Automatic music generation, as a compositional tool or autonomous system, continues to be a broadly compelling but challenging task. Many composers and researchers, using myriad techniques, have attempted creating musical pieces or fragments without direct human supervision. Success is often measured by aesthetic preference, of the composer or an audience, and achieving this typically requires a computational model with carefully crafted constraints. While highly tailored systems can produce good results in specific contexts, the general applicability to a wide range of musical styles is limited.

Human musicians learn to create and compose by listening to a lifetime of music and emulating preferred exemplars. In this way composers gain experience through mimicking other works and gain valuable knowledge about specific model pieces or songs. We propose a system that can operate in a stylistically agnostic fashion, working from examples to create new pieces that maintain a relationship to the sources.

Music composition practices differ from country to country, culture to culture. Recent advances in technology are creating a new era for music and music technology. With the help of modern technology and scientific theory researchers are trying to solve music composition problems automatically. Music composition requires extensive knowledge of music, music theory, and compositional technique. With the help of computers, many models of music composition systems have been developed and used.

Genetic algorithm(GA) is an optimization technique, which is based on the theory of natural evaluation. Genetic algorithm uses heuristic search and biological evaluation process to solve constrained and unconstrained optimization and searching problems. Automatic music generation using genetic algorithm deals with both searching and optimization problem. The main theme of using genetic algorithm in automatic music generation system is to generate music which is musically correct and the system takes less time to generate the music piece.

2. RELATED WORK

The automatic music generation problem has been solved in verity of ways by numerous researchers. Researchers try to develop automatic music generation system by using Markov chain, evolutionary algorithm, phrase imitation, genetic algorithm and neural network.

Researchers use the genetic algorithm to solve numerous problem of different fields. Researchers also try to deploy a new genetic algorithm and improve the performance of the existing genetic algorithms. Jin et al.[1] propose an improved genetic algorithm which uses three principles for selecting the schema, K-intensive effect synthesis operator and general five-intensive effect synthesis operator in the algorithm. Cerf [2] introduces another new genetic algorithm which uses two operator crossover-selection and three conditions of biological relevance in the crossover to improve the efficiency of the algorithm.

The genetic algorithm is one of the most commonly used technique by the researchers to compose music automatically. Each researcher uses genetic algorithm differently in their work to generate music piece. Paper [3] implements an artificial music composition system using genetic algorithm which can be used for any music genre, instrument and melody. By aiming to keep the melodies intact, the system uses the entropy of notes distribution as a part of the fitness function and the harmonic rules regulates the mutation and crossover function. Some researchers develop automatic algorithm system only for specific type of music. Paper [4] uses the genetic algorithm to generate melody only for Chinese folk music. Evolutionary melody operators are used to create new individual and Chinese folk music rules, in the form of fitness functions are used to evaluate the individual. Vargas et al. [5] propose a new genetic algorithm which uses music theory and smart operator to create artificial music patterns for Jazz music. Paper [6] proposes phrase imitation-based evolutionary composition(PIEC) to generate music using an evolutionary algorithm where PIEC regulates the intra-phrase and inter-phrase rearrangement to imitate the ascending/descending motion of phrases. PIEC is controlled by the four fitness function - note distribution, interval variance, and music theory. Another paper [7] applies predefined rhythm at the

initial population; position based representation of rhythm and relative representation of pitches while implementing genetic algorithm for automatic music composition. Özcan and Erçal introduce a genetic art tool named AMUSE, where the components of a genetic algorithm are embedded into it. Three intelligent fitness functions are deployed and the efficiency between standard fitness function and intelligent fitness functions are compared.[8]

Artificial intelligence has played a major role in many researches for developing a system that can composes music piece automatically. Researchers use pattern recognition, neural network as the part of artificial intelligence. Paper [9] introduces a system which uses chord grid and melody of a song to generate rhythmic accompaniment for guitar. Two artificial intelligence techniques are deployed - case-based reasoning and rule-based reasoning, to generate rhythmic accompaniment.

Some researchers try to implement Markov chain to build an automatic music composition system. Paper [10] implements an algorithm based on conditional probabilities which computes probability matrices of pitch and duration and two order Markov chain where first order of the Markov chain generates music piece has the flexibility of randomness and second order Markov chain has the constraint to generate music piece similar to original music piece. This system also gives the ability to the user to transform the phase such as pitch inversion and amount of deviation of the new music piece to the original one. Markov chain based web application named FlowComposer uses user specified lead sheet single polyphonic music and constrained based lead sheet generation tool to analyze and compose musical lead sheets[11, 12]. Another paper, [13] proposed an application based on four order Markov Chain and artificial intelligence to generate music melody. Note or pitch values, and a probability vector for each note is constructed in the first order of the Markov chain. To find the successive note in the melody, probability table is created by entering the frequencies.

A paper uses machine learning and artificial intelligence models to generate music piece. Recurrent neural network - a tool of artificial intelligence is used to produce notes. The machine learning tool - Naive Interleaving Approach is applied to generate melody and harmony of the song [14].

Another paper introduces a new music generation system named NetWorks (NW) which is inspired by Honing Theory of creativity. This system implements a hierarchically clustered scale-free network to generate music [15].

All of the above mentioned papers use single music piece as a basis to generate a new music piece. Moreover, most of them [3, 6, 9, 14, 15, 16] also use the MIDI file as the source to generate music piece. Some of the papers [16, 17] use conventional genetic algorithm but intelligent fitness function to generate the music piece. But this paper proposed a system which uses two MIDI files as a basis of generating a new music piece and it also proposed a new form of genetic algorithm for the generation of music piece.

3. PROPOSED SYSTEM

The proposed system uses user provided example MIDI files to guide the newly generated music pieces. The user specifies the length of the newly generated music piece as well as the weight of influence for each of the source examples. The automatic music generation system comprises two parts. The first part is analysis and information retrieval, and the second part is generating music using a newly proposed genetic algorithm. In the information retrieval phase, the musical features of two MIDI files are extracted. These musical features in turn guide the Genetic Algorithm to generate musical pieces. Figure 1 illustrates the overview of the proposed system for music generation.

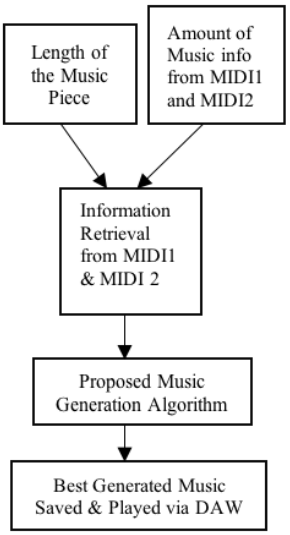

Figure 1. Proposed system.

4. AUTOMATIC MUSIC GENERATION

The automatic music generation algorithm has different phases, as follows.

4.1 Feature Extraction

In the feature extraction phase, different features are computed from MIDI note data. On the basis of midi velocity, a list of MIDI Note On events are compiled for analysis. Next, the number of appearances of each chromatic interval within an octave (unison, half step, whole step, minor third, etc.) are counted and used to compute a vector of percentages. After calculating the features for each MIDI file, the average is calculated across all files. The weight of musical features for each song selected by the user (in %) is specified, and used to guide the generation of a new musical piece. These features will be used to calculate the fitness. The highest and lowest Note numbers of each of the MIDI files are also calculated and the highest note of the two MIDI files and the lowest note of the two MIDI files determine the possible range of the generation process. Time event and Velocity event are processed from the two MIDI files and the mean of the Time event and Velocity event are likewise used in the automatic music generation step.

4.2 Create Parents (MIDI File)

500 parents are generated at the beginning of the genetic function, each consisting of a string of MIDI notes. The number of Note On events is the equal to the user input for piece length. Each parent is created by randomly selecting MIDI notes from the example files.

4.3 MIDI Data Set (Gene Set)

A MIDI Data set or Gene Set is created which comprises sequential Notes from the lowest Note of the two MIDI file to the highest note of the two MIDI files. This MIDI data set is used by the Mutation function to pick a random substitution Note while mutating a parent.

4.4 Fitness Function

Thirteen features of two example MIDI files are used to guide the generation of the new music piece. The total fitness of the new music piece is 100% and each music feature comprises 7.69% of the total fitness. For each music feature, fitness is tested based on the cumulative fitness of the two MIDI files and newly generated MIDI file. If the total fitness of the musical piece is less than 100 then the newly created fill will be mutated.

4.5 Mutation function

For 30% of the individuals, the mutation algorithm randomly chooses a note from the Gene Set and replaces one Note. The fitness of the new MIDI file is recalculated with this note replacement. The newly calculated fitness becomes the current fitness and the newly generated MIDI Note ON file will be saved as current MIDI File.

4.6 Crossover

The lowest 30% of the individuals are replaced with newly created strains using crossover. The crossover function randomly selects 30% of the individuals performs traditional single point crossover between these parents at random point. The newly crossover parents are added to the parent pool.

4.7 Number of Iterations

This system is developed in such a way that it can automatically choose the required number of iterations for optimal fitness. A good number of experiments are done to find out the number of iterations required to get the optimal fitness. At the initial stage, the number of iterations is set at 100. But if the best fitness increases within that 100 iterations, the count is reset, until 100 iterations pass without any best fitness improvement. 100 is chosen because experiments showed that for the proposed algorithm, if the increment does not occur within 100 iterations, it is extremely unlikely to occur in the next 4,000 iterations.

4.8 Algorithm

The paper proposed a new genetic algorithm to generate new music piece guided by two different exemplar songs. The algorithm for generating new music pieces using the genetic algorithm is as follows:

1) Generate 500 parents by selecting random MIDI notes from the example files and store them.
2) Pick 150 parents (30% of 500) from those 500 parents randomly.
3) Mutate each of the 150 parents for single time and replace the original parent with the mutated one.
4) Find the fitness of 500 parents (Original 350 and mutated 150).
5) Remove the 150 parents according to the low fitness.
6) Randomly select 150 parents from 350 parents and perform single crossover between them at random point.
7) Append all 150 newly crossover child in the parents pool. By appending these 150 child into the parent pool, the number of parent in the pool again goes to 500.
8) Go to step 2.

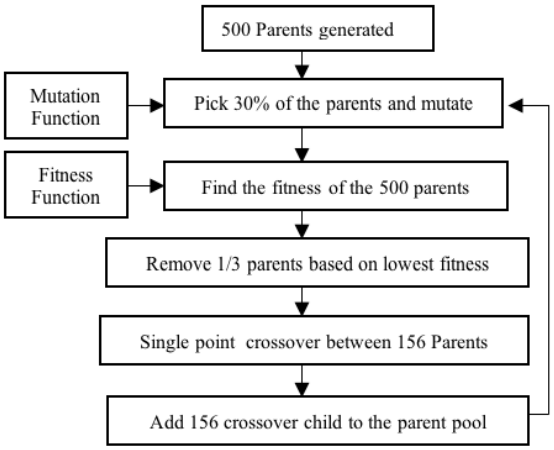

Figure 2. Proposed algorithm for music generation.

After getting the optimal fitness, the newly generated music piece is stored as a MIDI file. Figure 2 shows the overview of the proposed algorithm.

5. EXPERIMENTAL RESULT

The result of the newly proposed genetic algorithm is evaluated in three forms. First way is the computational efficiency of the system, second form is the time complexity and final way is musical analysis. Another system based on conventional genetic algorithm is also developed to compare the result and efficiency of the newly developed algorithm. This convectional genetic algorithm based system extract MIDI data from two MIDI files, generate two parents from the Note On range of the two MIDI files, crossover between them and calculate the fitness of both parents. The parent with highest fitness is regarded as parent and mutation is done on that parent. When the fitness is less than 100, the parent is mutated till 5,000 iterations. This conventional genetic algorithm based system uses all the same extracted musical features that are used in the proposed system from the same two music piece except the algorithm.

5.1 Data Set

For experimental purpose, ten MIDI music files are chosen to conduct the experiment.

MIDI File Name	Number of Note ON Events
Marry had a little lamb	6293
All of me	3390
Breathless	11334
Country road	12164
Ratlin Bog	4070
Jingle Bell	480
Always somewhere	8252
Emptiness	1296
Amay Proshno Kore	4388
Hum tum	8932

Table 1. MIDI Files used for experiment.

Two MIDI files from the above ten MIDI files of Table 1 are taken each time and variations of parameters are used. 500 music pieces are generated by using both proposed algorithm and conventional genetic algorithm to compare the computational efficiency and time complexity.

5.2 Computational Efficiency of New Algorithm

Based on the fitness function, the best fitness of the newly developed algorithm is about 88-92%. This result indicates that the proposed algorithm generates 44 - 46 notes correctly out of 50 notes. On the other hand, The efficiency of the system based on conventional genetic algorithm is 81-84%. On of basis of computational efficiency/fitness function, the proposed genetic algorithm is more efficient to generate musical notes than the conventional genetic algorithm.

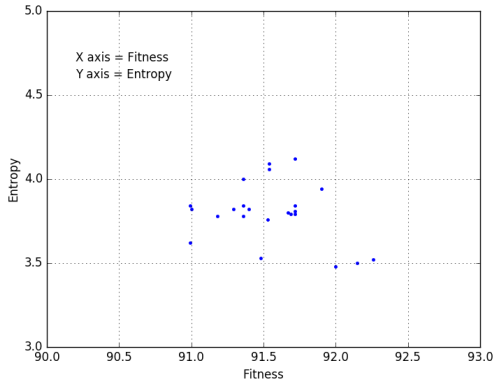

Figure 3. Entropy of newly Generated music pieces from All of Me and Marry Had a Little Lamb Song.

The entropy of first 50 Note On events of the All of Me song is 2.79 and the entropy of first 50 Note On events of the Marry Had a Little Lamb is 3.89. The entropy of all the generated music piece from these two MIDI files is from 3.48 to 4.12. Figure 3 shows the entropy of newly generated music pieces from All of Me and Marry Had a Little Lamb Song.

5.3 Time Complexity of New Algorithm

Time complexity of genetic algorithm mainly depends on the number of iteration need for getting optimal fitness.

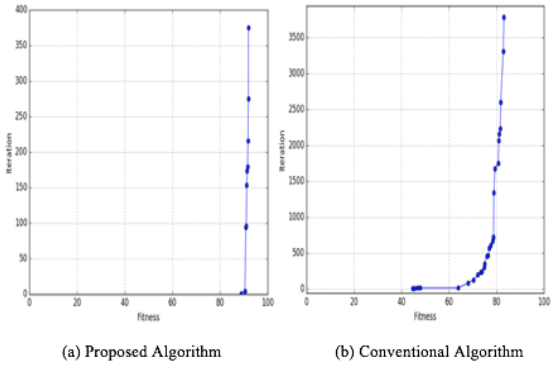

Figure 4. Fitness Vs. Iteration of proposed algorithm (a) and conventional algorithm (b).

Figure 4 (a and b) shows Fitness vs. Number of Iterations for the proposed algorithm and the conventional algorithm of generation a new music piece. In comparison with the traditional genetic algorithm, the proposed system needs less time to get the optimal fitness. To get the highest fitness using conventional genetic algorithm, the mean iterations to generate each music piece is 4401.16. In case of proposed algorithm, only 384 iterations needed to get the highest fitness for each music piece. The number of iterations also greatly affect the time complexity of the system.

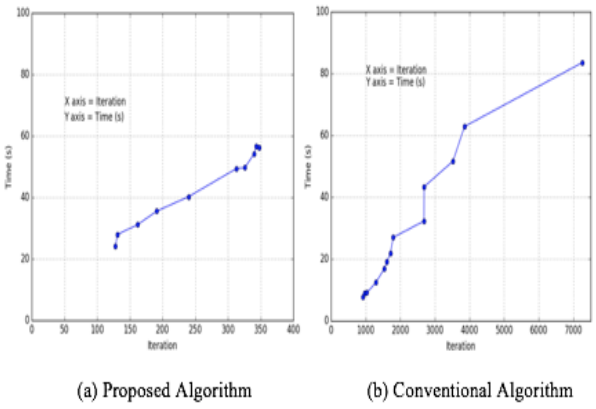

Figure 5. Iteration Vs. Execution Time (in second) of proposed algorithm (a) and conventional algorithm (b) for different music piece.

Figure 5 (a and b) shows different Iterations vs. Execution Time for the proposed algorithm and conventional algorithm of generation a new music piece. In proposed algorithm, the mean time needed for each iteration is 0.17 second. In the conventional algorithm, the mean time needed for each iteration is 0.07. But in the proposed algorithm, the mean of the number of iterations is 384 for getting the optimal fitness where mean 4,401.16 iterations are needed to get optimal fitness in case of the conventional algorithm which is much higher than the proposed algorithm. Thus, the lower number of iterations make the proposed algorithm more efficient than the conventional algorithm.

5.4 Musical Analysis of New Music Piece

From the result it is seen that for same song and same parameters, the newly generated music piece is different each time, but intervallically consistent.

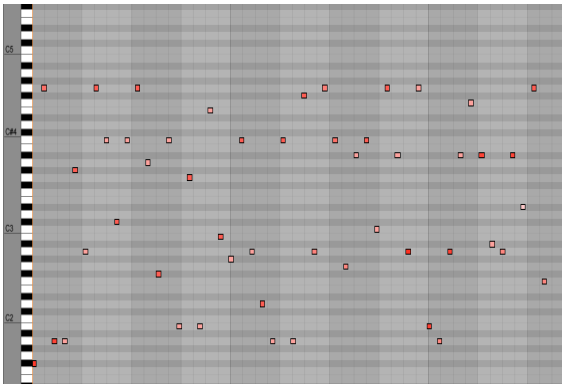

Figure 6. Piano roll of a generated piece using a target of 90% All of Me and 10% Mary Had a Little Lamb.

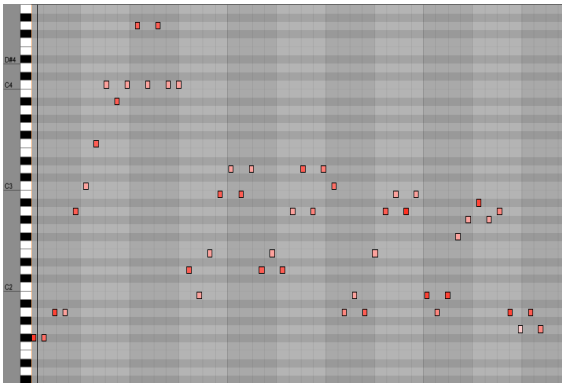

Figure 7. Piano roll of a generated piece using a target of 10% All of Me and 90% Mary Had a Little Lamb.

Figure 6 and 7 show randomly selected generated pieces. Weighting the fitness heavily (90%) towards one of the sources MIDI files produces results with figuration coherent with the original, yet distinctly different (it would be unlikely that a listener would be able to identify the source aurally). Figure 6 shows a lot of large interval skips, over a 3-octave range, with little musical motion or direction. Figure 7 is highly contrasting, with a distinct ascent followed by repeated figuration, gradually descending to the end.

The aim of this paper is to generate a music piece which will follow the weighting of musical features from each of the MIDI files to guide the generation. Figure 8 and Figure 9 show the Note sequence vs Note Number graph of two sources MIDI file All of Me, Marry Had a Little lamb and generated music piece by using these two MIDI files.

In Figure 8, the generated music piece uses 90% musical features from the All of Me song and rest 10% musical features are used from the Marry Had a Little Lamb song. The graph in Figure 8 shows that the generated music piece is mostly following the All of Me song.

In Figure 9, the generated music piece uses 90% musical features from the Marry Had a Little Lamb song and rest 10% musical features are used the All of Me song. The graph in Figure 9 shows that the generated music piece is mostly following the Marry Had a Little Lamb song.

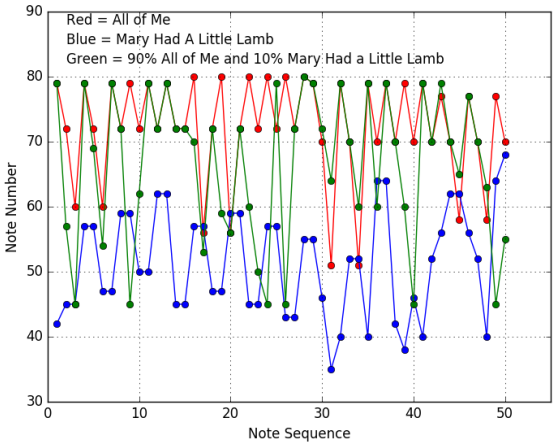

Figure 8. Relation between Exemplar Files (All of Me and Marry Had a Little Lamb) and generated file (90% All of Me and 10% Mary Had a Little Lamb).

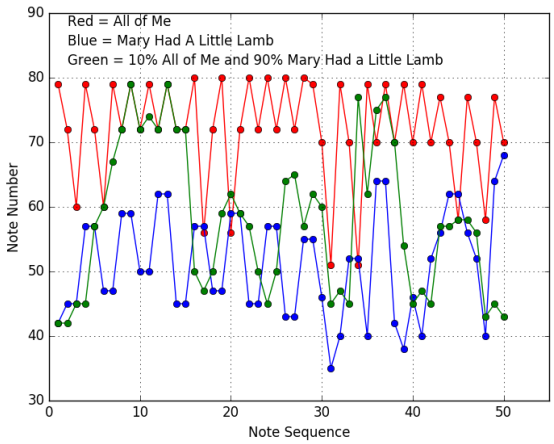

Figure 9. Relation between Exemplar Files (All of Me and Marry Had a Little Lamb) and generated file (10% All of Me and 90% Mary Had a Little Lamb).

6. FUTURE WORK

Beyond the scope of the current implementation, more features will be extracted from MIDI files by using different types of MIDI feature extraction framework, for example, Pulse [18], jSymbolic [19], music21 [20] to improve the aesthetic value of the newly created music piece. After generating music by using the features of each framework, results will be compared based on musical correctness and system efficiency. The best feature extraction framework will be chosen for the final music generation system.

7. CONCLUSIONS

This paper implemented an automatic music composition system which will expand the creativity of the composer. This system allows the user to choose the length of the music piece and how much the new music piece will be aligned with the two different music pieces.

Till now, most of the researches have been done to generate a new music piece by using a single music file. But in this paper, two music files are used to create a new music piece. Moreover, the user has the flexibility to choose the weight of influence for each of the source files. As a result,

the newly created music piece can be aligned with the two exemplar music pieces or highly aligned with one music piece and take less musical information from other music piece depending on the user's choice.

In the conventional genetic algorithm, one parent is created at the beginning and each time of iteration, single element from the parent is mutated. But in the proposed algorithm, 500 parents are created at a time and single element of each parent is mutated each time which improves the efficiency of the proposed algorithm as well as the system.

Getting optimal result with less time complexity and generating music piece with aesthetic value are still challenging in the field of automatic music generation and genetic algorithm. So far, many researches have been done to get the optimal result. But both the fields have the areas of improvement. These possibilities encourage proposing a new system for automatic music generation.

8. REFERENCES

[1] C. Jin, F. Li, M. WilamowskaKorsak, L. Li, and L. Fu, "BSPGA: A new Genetic Algorithm for System Optimization and Excellent Schema Selection," *The Annals of Applied Probability*, vol. 31, no. 3, pp. 337–352, 2014.

[2] R. Cerf, "A new genetic algorithm," *The Annals of Applied Probability*, vol. 6, no. 3, pp. 778–817, 1996.

[3] D. Daylamani-Zad, B. N. Araabi, and C. Lucas, "A Novel Approach to Automatic Music Composing: Using Genetic Algorithm," in *Proceedings of the 2006 International Computer Music Conference*, San Francisco, USA, 2006, pp. 551–555.

[4] X. Zheng, D. Li, Y. Z. Lei Wang, L. Shen, and Y. Gao, "Chinese folk music composition based on genetic algorithm," in *Proceedings of the 2017 3rd International Conference on Computational Intelligence & Communication Technology (CICT)*, Ghaziabad, India, February, 2017, pp. 1–6.

[5] F. V. Vargas, J. A. Fuster, and C. B. Castañón, "Artificial musical pattern generation with genetic algorithms," in *Proceedings of the 22015 Latin America Congress on Computational Intelligence (LA-CCI)*, Curitiba, Brazil, October, 2015.

[6] C.-K. Ting, C.-L. Wu, and C.-H. Liu, "A Novel Automatic Composition System Using Evolutionary Algorithm and Phrase Imitation," *IEEE Systems Journal*, vol. 11, no. 3, pp. 1284–1295, October, 2015.

[7] K.-C. Huang, Q. Jung, and J. Lu. (2007) Algorithmic Music Composition using Recurrent Neural Networking. [Online]. Available: https://web.stanford.edu/class/cs221/2017/restricted/p-final/quinlanj/final.pdf

[8] E. Özcan and T. Erçal, "A Genetic Algorithm for Generating Improvised Music," in *Proceedings of the 2007 International Conference on Artificial Evolution (Evolution Artificielle) EA 2007: Artificial Evolution*, Tours, France, October, 2007, pp. 266–277.

[9] D. Byrd, "Music Notation Software and Intelligence," *Computer Music Journal*, vol. 18, no. 1, pp. 17–20, Spring,1994.

[10] A. Antoine and E. Miranda, "A User-Centric Algorithmic Composition System," in *Proceedings of the 2016 International Conferences on New Music Concepts (ICNMC)*, Treviso, Italy, March, 2016.

[11] F. Pachet and P. Roy, "Imitative Leadsheet Generation with User Constraints," in *Proceedings of the 2014 21st European Conference on Artificial Intelligence (ECAI 2014)*, Prague, Czech Republic, 2016.

[12] A. Papadopoulos, P. Roy, and F. Pachet, "Assisted Lead Sheet Composition using FlowComposer," in *Proceedings of the 2016 22nd International Conference on Principles and Practice of Constraint Programming CP 2016*, Toulouse, France, 2016, pp. 769–785.

[13] S. Hill. (2011) Markov Melody Generator. [Online]. Available: http://www.cs.uml.edu/ecg/index.php/AIfall11/MarkovMelodyGenerator

[14] S. Bell and L. Gabora, "A Music-generating System Inspired by the Science of Complex Adaptive Systems," in *Proceedings of the 2016 The Fourth International Workshop on Musical Metacreation (MUME*, Paris, France, 2016.

[15] M. Dahia, H. Santana, E. Trajano, G. Ramalho, and G. Cabral, "Using patterns to generate rhythmic accompaniment for guitar," in *Proceedings of the 2004 International Conference on Sound and Music Computing (SMC) 2004*, Paris, France, 2004.

[16] D. MATIĆ, "A Genetic Algorithm for Composing Music," *Yugoslav Journal of Operations Research*, vol. 20, no. 1, pp. 157–177, 2010.

[17] J. Cooper and C. Hinde, "Improving Genetic Algorithms Efficiency Using Intelligent Fitness Functions," in *Proceedings of the International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems: Developments in Applied Artificial Intelligence*, Loughborough, UK, June, 2003, pp. 636–643.

[18] J. Langhabel, R. Lieck, M. Toussaint, and M. Rohrmeier, "Feature Discovery for Sequential Prediction of Monophonic Music," in *Proceedings of the 20017 18th International Society for Music Information Retrieval Conference (ISMIR 2017)*, Suzhou (China, October 2017.

[19] C. McKay and I. Fujinaga, "jSymbolic: A Feature Extractor for MIDI Files," in *Proceedings of the 2006 International Computer Music Conference*, Louisiana, USA, 2006, pp. 302–305.

[20] M. S. Cuthbert, C. Ariza, and L. Friedland, "Feature Extraction and Machine Learning on Symbolic Music using the music21 Toolkit," in *Proceedings of the 20011 12th International Society for Music Information Retrieval Conference (ISMIR 2011)*, Florida, USA, October, 2011.