AUTOMATIC MUSIC COMPOSITION USING GENETIC ALGORITHM AND ARTIFICIAL NEURAL NETWORKS

Iyad Abu Doush[1], Ayah Sawalha[2]

[1]Computer Science and Information Systems Department, American University of Kuwait, Salmiya, Kuwait

[1,2]Computer Science Department, Yarmouk University, Irbid, Jordan

Email: eyad_win@yahoo.com[1], ayah_sawalha@hotmail.com[2]

DOI: https://doi.org/10.22452/mjcs.vol33no1.3

ABSTRACT

The aim of this paper is to automatically compose new pleasing music from randomly generated notes without human intervention. To achieve this goal, Genetic Algorithm was implemented to generate random notes. The Neural Network was trained on a set of melodies to learn their regularity of patterns and then it is used as a fitness evaluator for the generated music from the Genetic Algorithm. Four Genetic Algorithms (using different combinations of tournament, roulette-wheel selections and one-point, two-point crossovers) were used in generating music to compare them according to which one is the most suitable for music composition. The experiments show that using tournament selection and two-point crossover produces better music patterns than using other combinations by 57%. The experiments show that the generated music was good and the results were promising. For evaluation, 10 music experts were asked to listen and evaluate four samples of the generated music; two of them were evaluated high from the Neural Network and two were evaluated low. Then we compared their results with the results from the Neural Network. The results show that the error rate for Neural Network was 16.7% and accuracy was 83.3%.

Keywords: Genetic algorithm, Neural network, Automatic music composition.

1.0 INTRODUCTION

Music is the Language of the Universe. Everyone can understand it and react to it in a certain way. Composing music is not an easy task, especially for people who do not have a musical background. As any other language, music has its own concepts and rules. Music theory had specified those concepts and rules. Such as: harmony, rhythm, melody, etc. and made it easier for composers to read and play music.

Music composition is the art of creating and innovating new music using the definitions and rules of the music theory. Several methods were used to develop music composition systems. Such as: Grammars, Knowledge-Based systems, Markov-Chains, Artificial Neural Networks, Cellular Automata, Evolutionary and Other Population-Based Methods [1, 2]. The idea of automatic music composition could be deployed in several domains. For example, it could be used as background music for movies or as mobile ringtones. It also could be used in music classes for teaching students musical styles like jazz and blues or teaching them the musical scales. Also, it could be used for developing listening skills for beginner musicians.

Automatic music composition is a very interesting field in Artificial Intelligence (AI). From the mid-twentieth century, many optimization algorithms were implemented to compose music [3]. As [4] described, Genetic algorithm is one of the best optimization algorithms, which has been used in this area. It is inspired from biological concepts such as mutation and natural selection. As described in [4, 5], the composed music parts either survive or die with respect to the selection following a Darwinist fashion, since few changes to key notes in a piece of music may change it to less interesting music.

Music theory is a set of rules that is used by all Western instruments [6]. As defined by [7], music theory is the understanding of the language of music. It is a way to understand the music that we hear. The concepts and rules of the music theory is similar to the grammatical rules of any written language. It helps in reading music as its composer meant it to be. However, a composer is not restricted with its rules. A composer may use the rules as a guide [7].

Music theory includes considerations of melody, rhythm and harmony. With enough knowledge of music theory and a familiarity with the techniques and languages of instruments, a composer can compose melodies. These melodies emerge from the possibilities within scales, modes, keys, techniques and limitations of the musicians who will be required to play them [7]. Music can be divided into two main types: Monophonic and polyphonic.

Monophonic means music with a single part or a single melody played on a single musical instrument. **Polyphonic** means music with multiple parts, where simultaneous notes are played at the same time from different musical instruments [1].

A piece of music consists of a sequence of notes with information about them. Such as: pitches (sound of notes), duration, interval and tempo. Notes can be named by the letters 'A' through 'G'. Notes can also have different durations. Where duration is how long a note lasts. The most basic notes durations are: Whole, half, quarter, eighth, sixteenth, and twenty third. A note may be dotted (dotted notes are also called sharp notes). A dot can be added to any note and then the note receives its original value plus half of it [5]. In total Western music uses twelve notes: A, A#, B, C, C#, D, D#, E, F, F#, G, and G#.

Within music there are rests (pauses). There are no notes played during the length of the rest. The values of rests are the same values of notes [5]. Tempo is the speed of the beat, usually given in beats per minute (BPM). Interval is the distance between two notes. It can be either harmonic for notes that are played simultaneously or melodic for notes that are played sequentially. Octave is a perfect interval. It refers to the interval between one note and another note with the same name with half or double its frequency. For example, from the note A up to the next A is one octave (A-B-C-D-E-F-G-A) [5, 7] as shown in Fig.1.

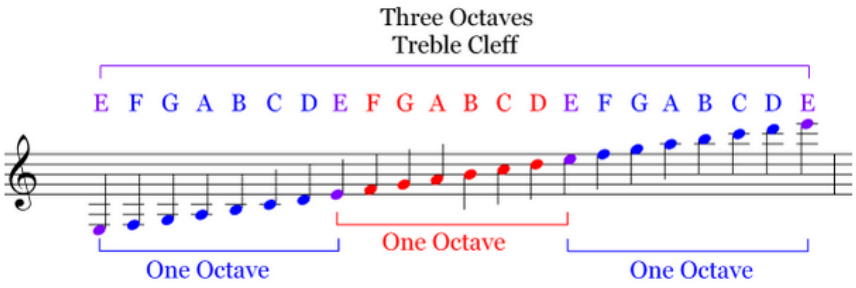

Fig.1: Musical Octaves [8]

Genetic Algorithm (GA) was invented by John Holland in 1975 [9]. The idea of it was to exploit the idea of evolution techniques such as selection, crossover and mutation to solve optimization problems [10, 37]. It randomly generates an initial population, computes and saves the fitness for each individual in the population. After that, selects best individuals and perform crossover which represents mating between individuals and Perform mutation which introduces random modifications. It repeats these steps until optimal solution is obtained or maximum number of generations is reached [38, 39]. There are 3 types of fitness functions for Genetic Algorithms used in music composition, which are:

(1) Interactive fitness functions. Where the human acts as the fitness function. As he listens to each musical output and give it a rate. This approach is very time consuming since listening to every single output and then rate it generates a bottleneck.
(2) Knowledge-based fitness functions. Here the fitness function is designed based on the rules of music theory. This approach requires prior knowledge to music rules. And it restricts the music composition to a single music style.
(3) Machine-learning based fitness functions. In this approach a machine learning technique is used in order to learn regularities of patterns from music samples [11, 12]. These types of fitness functions are the best choice for people who do not have musical background. In addition, it will not be restricted to any musical style.

Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the biological nervous system. It consists of a large number of highly interconnected processing elements called neurons or processing elements, working together to solve specific problems [2]. ANN consists of three basic layers: input layer, hidden layer and output layer. Hidden layer may consist of more than one layer. Each processing element is fully connected

with all processing elements in the previous layer. ANNs learn by example. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning in ANNs involves adjustments to the connections that exist between the neurons until the desired output (ideal output) is achieved [2].

In this paper, we develop an algorithm that is capable to automatically compose music melody patterns using Genetic Algorithm (GA) with an Artificial Neural Network (ANN) as a fitness evaluator. ANN was trained on a set of melodies to learn their patterns to work as a fitness evaluator for the generated music. In the suggested scheme, the GA generates chromosomes of randomly generated notes which are evaluated by an ANN trained on a set of music patterns in the RTTTL format form a chosen musical style. The generated notes could be directly used by musicians or used as an inspiration for them. This would reduce time and cost for rapid music production, beside the benefits of its commercial applications. Several music composition algorithms have been produced. Each algorithm had focused on some styles of music such as jazz and rock. In this paper, we will not focus on one style. The system is capable to learn any musical style and compose melodies according to it. The algorithm composes short monophonic melodies since polyphonic melodies consist of multiple levels of features.

2.0 RELATED WORK

Music theory [7] is the grammars and rules that define how musicians and composers compose music. It helps read and understand music. Music theory includes considerations of melody, rhythm, counterpoint, harmony, tonal systems, scales, tuning, intervals, consonance, dissonance, durational proportions and the acoustics of pitch. All of the early computer music works were relied on the relationship between music theory and mathematics. In 1951, CSIRAC (Australia's first digital computer) was the first digital computer that played digital music in the world. The system was programmed by the mathematician hill to play some of the early 1950's popular musical melodies. Its' music was never recorded. Another computer program was written by Strachey in 1951 to generate music. The program recordings were played by Manchester Electronic Computer (the world's first commercially available general-purpose electronic computer) [13].

In 1950's, algorithmic compositions and digital sound synthesis by computer started. Credit goes to two major developments: "MUSIC I" and "Illiac Suite for string Quartet". Max Mathews wrote the first real music program. He called it MUSIC I. It played a single line Tune. It was followed later by different enhanced versions [14]. As mentioned in [10] [15], one of the earliest examples of composed music using computers was the "The Illiac Suite for String Quartet" by Hiller and Saacson (1958). It was programmed to generate random sets of integers representing notes with different musical elements, such as pitches and rhythm.

Evolutionary algorithms (EAs) are suitable for generating solutions for optimization problems. They are inspired by Darwin's biological model of evolution and natural selection. They apply the principles of evolution found in the nature to the problem and trying to find an optimal solution. EAs are based on biological mechanisms, such as: selection, reproduction and mutation. At the beginning, they start with a set of candidate solutions. Some candidates are selected according to their fitness function value. Crossover and mutation operators are applied on candidates. These steps are applied iteratively until an optimal solution is found [2].

Matic [5] presented an approach to compose music using Genetic Algorithm. His aim was to produce short pleasant compositions. The goal of the approach was to exploit all the sets of all individuals rather than optimization. The individuals of the initial population had some predefined rhythms, to give good starting solutions. The crossover operation was omitted and instead of that three types of mutations were used and applied on the best individuals multiple times to make them better. The used fitness function is based on different criteria of measurable musical elements (pitches, notes durations and tonality). The generated music is evaluated by computing the similarity with the reference individuals. The approach resulted compositions with some pleasant intervals and a meaningful rhythm. Results showed that the combination of a large number of different parameters can significantly affect the quality of the generated melody. This approach was somehow rigid. Omitting the crossover operator limited the approach convergence. There is no expert evaluation of the generated music.

One of the most interesting software models in the automatic music composition field was implemented by Biles [16]. This model was called GenJam. It was developed using Genetic Algorithm with an interactive fitness function. Each individual is evaluated by a user who listens to the resulted music and gives it a fitness value. This application is used for Generating a Jazz Solos. The GenJam had two populations instead of one. The first population was the measure population and the second one was the phrases population, where each individual of the phrases population

maps to indices of measures in the first population. Both populations were used to build a solo. This version of GenJam had two disadvantages. The notes occurs only in eighth note multiples and only 14 pitches were available to choose from. The method evaluation is limited as it can only check the music melodies generated from the phrases population.

Khalifa et al. [17] had developed an evolutionary music composition system using Genetic Algorithm with two tonal fitness functions. The system consisted of two stages of evolution processes. At the first stage, short motifs (musically sound patterns) are evolved using interval evaluation fitness. At the second stage, two evaluation functions were implemented intervals and ratio. The ideal ratios of the notes are predefined by the user. In their work they have chosen 60% for the notes that make up the chords within a key, 35% for the notes remaining in the key and 5% for the notes which are outside the key. The generated music was translated into Guido Music Notation (GMN). The approach was able to create interesting music that is innovative and with a pleasing musically sound. The use of interval and ratio limited the music evaluation as it would be not possible to use for evaluating other types of music types.

Oliwa et al. [12] developed a knowledge-based evolutionary music system. It had a built-in knowledge in the structure of the composed music. It was based on Time Delay Neural Networks (TDNN) and Ward Nets on a probabilistic finite state machine (FSM). Their system used Genetic Algorithm and the "abc" music representation to evolve four types of music: lead guitar, rock organ, drum and rhythmic guitar of structured rock music. There were 5 Genome types with different integer configurations. At each run, a randomly Genome structure set was generated where each Genome is specified for an instrument and had a set of fitness functions. The training data consist of 19 songs written in "abc" language which is translated into MIDI. The results from the FSM noted that the overall quality of the songs did not change over time in contrast to the NN songs. The TDNN-Ward Net was trained over 72000 epochs with an error of 0.0006718 on the training set. The results showed that the NN was oscillating more often between extreme notes with large intervals between them. The NN was not able to memorize the whole song.

Manaris et al. [18] presented a corpus-based hybrid approach to analyze and compose music. It is a mix of 3 components: statistical, connectionist and evolutionary. The approach identifies the similarity between the resulted music and a set of favourite songs. Genetic evolutionary algorithm was used to generate music. To evaluate their approach, they had trained artificial neural networks (ANN) to predict a popularity of 992 music pieces. They examined their work in 3 ways. First way, using music features based on zipf's low. The second way, using a genetic- programming system called "NEvMuse". The last way was the critics from 32 human subjects. The results were as follows: in the first test, ANN accuracy was 78.5%. The ANN rate in the control test was 49.68%. In the third test, ANN accuracy was 86.11%. The results obtained show that there is still a room for enhancement of the generated melodies.

Chen [19] presented Elman's Neural Networks to generate melodies. Genetic algorithm was applied on the top of the Neural Networks to maximize the chance of producing good melodies. Chen presented a melodic constraint (on tonality and rhythm) to determine the fitness function. Five musical notations are used (whole note, half note, quarter note, eighth note and sixteenth note) and 3 octaves (c2 to c5). The used constraints are for the composing style and for the rules of the music theory. In total 7 constraints were used. A constraint could be turned on or off. A constraint may dominate other constraints by giving it a higher fitness score. The optimal result in the first experiment came after 750 generation and in the 2nd experiment after the 200th generation. The several rules used in the constraints make the algorithm convergence slower and the large numbers of constraints to switch make it more difficult to tune the generated music.

Sánchez et al. [20] developed a tool called "Spieldose" or the music box, using interactive Genetic algorithm. The initial population of the algorithm parents is selected by the users. The melody is represented as a sequence of notes. Each note has its pitch, length and type. The contribution in this approach is that they have used different types of crossover, mutation and improvement operators. It permits the automatic correction of some musical errors which could be caused by the previous operators as it applied the rules of the music theory. This technique needs the user input to provide the initial population.

Sheikholharam et al. [21] had used Genetic algorithm and Kohonen musical grammar to compose music melodies. They had followed the MIDI standard of assigning the value of 60 to the middle C. They used 6 musical durations: 32 sec notes, semi-quavers, quavers, crotchets, minims and semi-breves for duration. The Kohonen musical grammar is used to extract musical rules and patterns to form the fitness function used to evaluate the generated melodies. The generated melodies are not checked as there is no evaluation provided for the proposed algorithm.

Muñoz and Ong [33] propose a technique for automatic composition of music based on memetic approach that uses fuzzy decision trees. The algorithm relies on utilizing multi-agent memetic composers to generate melodies using harmony rules. The algorithm takes a bass line as an input and produces a high quality piece of music by optimizing the melody. The obtained results show that proposed memetic approach statistically overcomes other evolutionary approaches used in the area of music composition. The evaluation is based on listening tests in which people with music backgrounds listen and compare between the melodies generated by the proposed algorithm with the melodies composed by other algorithms.

Ting et al. [34] introduce the phrase imitation-based evolutionary composition (PIEC) for automatic music composition using genetic algorithm based on music theory. The PIEC rearrange the music phrases to simulate the phrases ascending or descending change. The authors developed four fitness functions for the composition progress by considering note distribution, interval variance, and music theory. The evaluation of the generated melodies provides evidence that PIEC is able to generate melodies that hold the input melody characteristics. The results point out that the four fitness functions used are effective in generating melodies that have the same distribution of notes as the initial sample melody.

Based on the literature, few researchers tackle develop an end-to-end solution for automatic music composition with a system evaluation validation. We propose a novel algorithm to compose music and evaluate the generated music using artificial neural network. After that, we compare the obtained evaluation with human expert evaluation.

2.1 Composing Music Using Other Techniques

Many techniques from the AI field are used to compose music beside the evolutionary algorithms. The most common techniques are: Grammar-based systems, where a set of rules are adopted from the music theory used as a formal grammar. Another technique is Knowledge-based systems, which use some static composition rules. Moreover, Markov-chains technique is used, where the probability matrix could be derived from a training corpus of existing music or derived by hand from the music theory rules [2].

Eck et al. [22] built a model to compose music based on LSTM (long short-term memory) Neural Networks. They built this model to overcome the problem of the missing global structure in recurrent neural networks (RNN). Where the networks are not capable to learn the entire musical form, and use its knowledge to guide the composition procedure. They provided a comparison between data representation in their model and the previous ones. They represented the data in a local form unlike the other model that had left it to the Network to learn it. The training data was a 12-bar of blues music that is popular among bebop jazz musicians. There were 8 notes in each bar, and no rests were included. Their results showed that the LSTM model had learned successfully music patterns (blues music) and was successfully able to compose new music. However, the obtained results are not validated with a human expert.

Hörnel et al. [23] implemented a system called MELONET II. It uses a combination of neural networks to generate new music. Several neural networks were combined to learn and classify the attributes of a song at different time scales. The system is specialized in composing folk music. It gives a short part of a melody to begin with. The used Neural Network had 3 layers (input, output and hidden). They tested the program with 20 folk songs and compared the results with the results of a reduced version of the model which consisted only with a subnet that is equivalent to a simple standard network. The network model was able to learn all the songs whereas the standard network could not learn them all. The results were better coherence compared to previous systems as mentioned by the authors. The proposed system does not have a method to validate the pleasing sound of the generated melodies.

Morris et al. [24] developed a commercial system called “MySong”. It automatically generates a set of chords to accompany the voice of an end-user who sings to a microphone. The system was developed using the Hidden Markov Model (HMM). The chords form the hidden states and the observation states are represented in histograms. The system pre-processes the training data into musical key of C by transposition, thus the key signature must be given or identified in advance. The drawback of this system is that it will not recognize any note that is not specified in the pre-process. HMM is only capable to capture patterns of short musical pieces.

Monteih et al. [25] described a system that automatically generates and evaluates musical accompaniments for a specified set of lyrics. The used accompaniments were in 3 categories: nursery rhythms, folk songs (bluegrass) and rock songs (beetles). An n-gram model was used to generate pitches. It was constructed from melodies of similar

song styles. The system analyzes all the lyrics text and then assigns it a rhythm. The results showed that the system was able to generate pleasing melodies that fit well with the lyrics text.

Chan et al. [26] presented a system called ACSSM II. It generates music by searching for a sequence of music segments that best satisfy different musical constraints, such as: pitch length and range, harmonic backbone and consistency using a probabilistic model of a computer style. The system exploits an input corpus of music, genetic algorithm and Markov chains as machine learning techniques. They have used 1-point crossover, and 5 mutation functions as follows: prepend a new segment to a sequence, append a new segment to a sequence, remove a front segment from a sequence, and remove an end segment from a sequence and replace the sequence with a randomly generated sequence. The used fitness function was the sum of two heuristic measures: transition probabilities and the transpositional displacement. The generated melodies are evaluated by three experts who listened to the output music and give it a score in the range 0-10. The scores mean was 7.34.

Opolka et al. [35] apply Answer Set Programming (ASP) technique to compose music, the proposed technique is called chasp. The ASP is used by proposing a set of rules based on the theory of harmony. The resulted harmonic sequence can be used as the seed to create simple melodies. In another work, Sfu and Yang [36] propose a methodology to build an automatic music transcription dataset. The proposed dataset is not restricted to one class of instrument. The dataset feasibility is verified by evaluating piano solos and four woodwind quintets. The results show that the proposed annotation is reliable to evaluate the automatic music transcription.

3.0 METHODOLOGY

The general idea of our work is to prepare a set of training music, feed it to the Neural Network in order to learn it. After the training, the NN will be used as the fitness function of the Genetic Algorithm. The Genetic Algorithm generates random notes, perform on them different selections, crossovers and mutation. Evaluate results using Neural Network. Fig.2 presents a flowchart that illustrates the idea of our methodology.

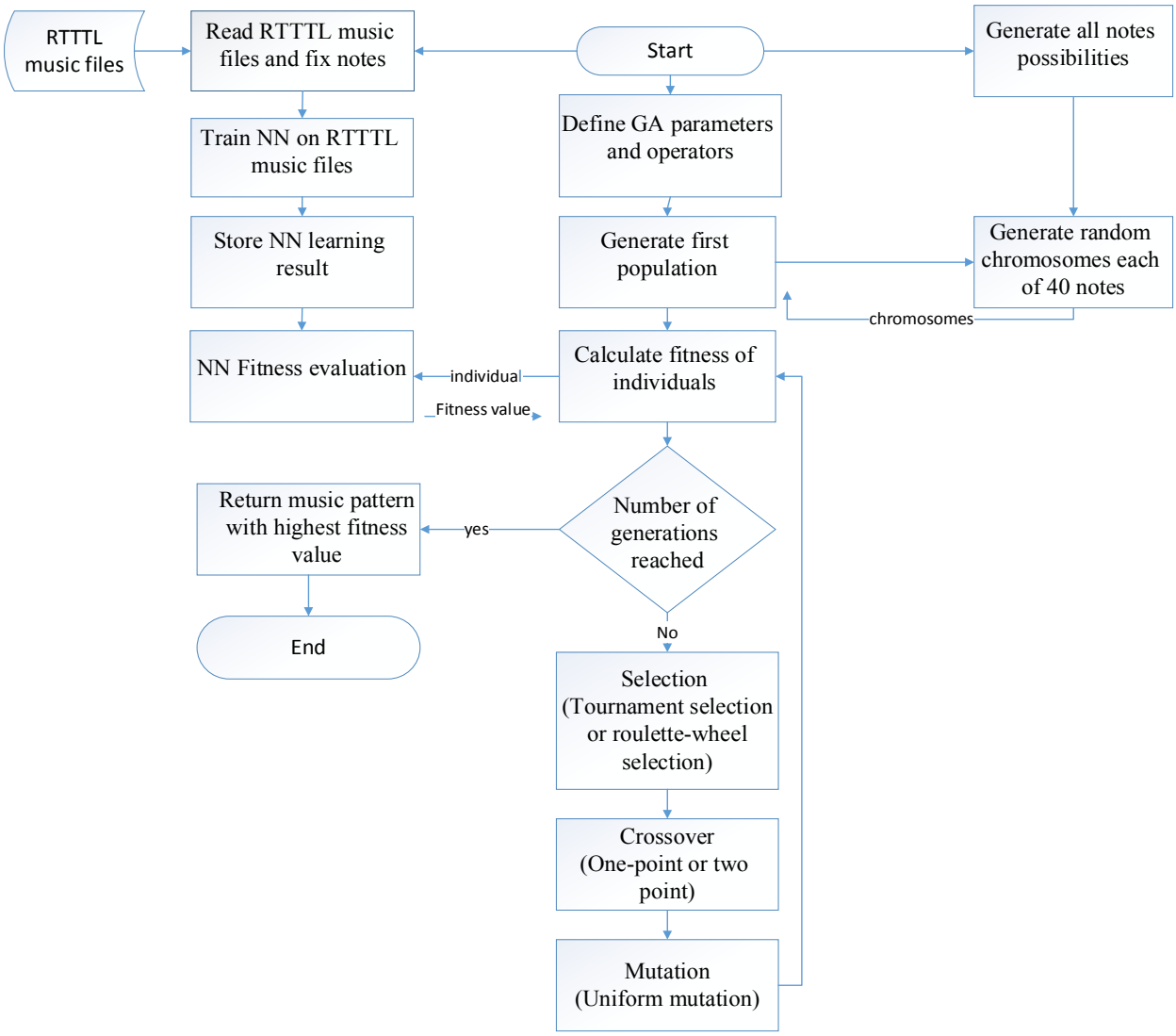

Fig.2: Flowchart of the methodology

3.1 Training Set

We used a set of 392 monophonic music files in the RTTTL format. RTTTL stands for Ring Tone Text Transfer Language. It is a language developed by Nokia. It allows someone to create his own ringtone using a series of commands that produce any number of different synthesized sounds [27]. We used RTTTL music format because the music patterns contain all the information about notes that we may need in our work.

RTTTL uses the 'abc' notation of notes. The pitches (pitches here means notes) that can be used in a RTTTL string are: A, A#, B, C, C#, D, D#, E, F, F#, G, G# and P (pause). A RTTTL melody is divided into 3 parts separated using a colon:

(1) The title. It contains the title of the melody. It may have at most 10 characters.
(2) Default parameters, which are: default duration (d), default octave (o) and default beat per minute (b) or as called "tempo". These parameters are assigned to notes which have no duration or octave values.
(3) Duration may be: whole note (1), half note (2), quarter note (4), eighth note (8), sixteenth note (16) or thirty-second note (32).

Octave starts from the 'A' note below middle C and may go up to four octaves. The actual melody notes consist of a set of character strings separated by commas, where each string contains: duration, pitch and octave. For example, a note may be written as: 8c#6. Where 8 is the duration, c# is the note and 6 is the octave. Or a note may be written as c#. In this case, the duration and octave values are taken from the default parameters section. We generated all

possibilities of notes along with their durations and octaves, and stored them in an array.

- We have 12 notes, which are: A, A#, B, C, C#, D, D#, E, F, F#, G, G# in addition to the pause P.
- 4 octaves, which are: 4, 5, 6, 7, and 8.
- 6 durations, which are: 1, 2, 4, 8, 16, and 32.

By multiplying them, we get 312 possible notes. Any generated note has 312 possible values. Suppose that we wanted the composed melody to have 40 notes. This means that we have a probability of 31240 which equals: 5.8369203e+99 of possible melodies that could be generated, which is a very huge number.

3.2 Neural Network

In music, there are good melodies and bad melodies. The bad melodies are unlimited. Therefore, the ANN will fail to classify a new melody to good or bad melody if we did not provide it with all bad melodies, which is impossible. So, instead of feeding the ANN with good and bad melodies, we separated the good melodies (training melodies) into 2 sets of the same music type. The first set is set1 and its ideal output value was 1. The other set is set2 and its ideal output value was 0.

In order to consider whether a melody is good, its fitness value must be close to 1 in set1 or close to 0 in set2. A bad melody will be neither close to set1 nor set2. After reading the two sets of training RTTTL files and assigning them their ideal values, the system read the notes of the RTTTL files. At this point the ANN is ready to start learning. We used the back propagation algorithm in the ANN training. It is a common method of training ANN. It works faster than other learning approaches.

The back propagation algorithm trains a given feed forward multilayer neural network for a given set of input patterns with known classifications. When each entry of the sample set is presented to the network, the network examines its output response to the sample input pattern. The output response is then compared to the known and desired output (ideal) and then the error value is calculated. Based on the error, the connection weights are adjusted [28] [29].

We also used the Activation Sigmoid activation function in the ANN and allowed the Bias option. Activation Sigmoid function is a bounded differentiable real function that is defined for all real input values and has a positive derivative at each point. The Use of additional input component, which is the bias improves properties of the neuron. It allows moving the threshold of activation function. Use of bias increases calculations, because the additional weight has to be determined [29]. After the Neural Network had finished training, the result was saved, to be used as the fitness function of the Genetic algorithm.

3.3 Genetic Algorithm

Several techniques exist to create computer generated music. One of those techniques is Genetic Algorithms. It is a suitable approach to compose music since its operators enable continuous enhancement of the generated music as what happens in real composing [5]. Besides that, because of the diversity of melodies over a specific range of notes is very large, genetic algorithm is a good choice to help in composing melodies.

We used two types of selection; Tournament selection with k=2 as recommended by [30] and Roulette wheel selection. Note that, in these two selections we use the variation in which we perform selection on the best half of the population. Two types of crossovers: one-point crossover and two-point crossover. The uniform mutation is used because it is suitable for genes presented in double values. Mutation rate was 0.1 and crossover rate was 0.8. These rates are obtained from [12]. The proposed GA works as follows:

- At the beginning, the configuration of the operators and the parameters of the GA is defined (mutation rate, crossover rate, number of generations, population size, type of selection and type of crossover)
- The GA starts by randomly generating the chromosomes of the initial population. Each chromosome consists of 40 random genes (notes).
- Rank the population. Compute the fitness value for each chromosome in the population and order them according to it. Then store the best 50% of them. Fitness is computed by sending chromosome value to the network function of the ANN which computes the fitness according to its similarity with music training set.

- Create next generation. By performing the following operations repeatedly until the new population is complete.
- Select two parent chromosomes from the population according to their fitness (the better fitness, the higher chance to be selected). Selection is performed according to a specified rate. We used one of two types of selection; Roulette Wheel selection or Tournament selection.
- Perform crossover to generate new offspring. One of two types of crossover was used; one point crossover or two point cross over.
- Mutation. Select random gene (note) in a chromosome and mutate it with a new gene (note).
- Add new offspring to the population. Repeat the previous steps starting from ranking the population until maximum number of generations is reached.

The following is the proposed GA pseudo code:

Input: crossover rate (cr), mutation rate (mr), population size (p), generations (G), Genome size, crossover type, selection type
Output: sequence of notes

```
//Initialization
 For I =1 to p do
   Generate randomly chromosomes each of 40 genes;
End for
rnd = U(0,1)
For j= 1 to G
   Compute fitness for each chromosome
   Rank chromosomes according to fitness
   If (selection type ==Tournament selection)
          Select fittest 50% of chromosomes
          else
           Roulette- Wheel selection of fittest 50% of chromosomes
    Save them in the population Pj;

 // crossover
 If  rnd < cr do
     Randomly select two solutions X1 and X2 from Pj;
     If (two-point crossover)
              generate X3 and X4 by Two-point crossover to X1 and X2;
            else
               generate X3 and X4 by one-point crossover to X1 and X2;
   save X3 and X4 to Pj++ ;
 End for

 // Mutation
 If   rnd < mr do
    select a solution X1 from Pj++;
    mutate X1 under the rate mr and generate a new solution X2;
    pj+=X2
 End for

End for
//Returning the best solution

return the best solution X in P ;
```

3.4 Implementation

Microsoft visual studio 2010, C# language was used to implement our work. We used the Encog framework [28] in implementing the Neural Network. Encog is a machine learning framework. The data sample were 392 music file of type RTTTL. It was downloaded from: http://www.picaxe.com/RTTTL-Ringtones-for-Tune-Command/.

The implementation consists of three phases. In the first phase, all notes possibilities are computed, to be used when the GA generates random notes. It selects randomly from them. Then the system reads the RTTTL music files and fixes their notes. Nodes need to be fixed because some of them are missing the duration and octave values, which are taken from the default parameters section. They are fixed by assigning them the default values. In the second phase, the ANN is trained on the RTTTL music files.

The training process happens only once. Then the training result is stored. In the third phase, the GA randomly generates a sequence of notes, applies on them different types of selection, crossover and mutation operators and then the generated music sequences are sent to the NN to evaluate their fitness value until maximum number of generations is reached. The output is a sequence of notes in the RTTTL type. In order to listen to the resulted music they are converted to MIDI format using Quick Ringtone tool [31].

4.0 EXPERIMENTS AND EVALUATION

We had two types of experiments. The first one was to evaluate four variations of GA according to the ANN fitness evaluation. The second was to evaluate the goodness of the generated music patterns according to music experts and evaluate the accuracy and error rate of ANN fitness evaluation.

4.1 GA Variations Evaluation

Four variations of GA have been evaluated: roulette wheel selection with one-point crossover, roulette wheel selection with two-point crossover, binary tournament selection with one-point crossover, and binary tournament selection with two-point crossover. We repeated each one of the four experiments 30 times. The GA setup used in this experiment is shown in table 1.

Table 1: GA Setup

Number	Operator/ Parameter	Value
1	Crossover Rate	0.8 [12]
2	Mutation Rate	0.1 [12]
3	Population Size	300
4	Generations	50000
5	Mutation	Uniform
6	Chromosome size	40 notes

Table 2 illustrates the fitness evaluation results. The values closer to one mean better fitness value. The highest fitness value in each row is in bold.

Table 2: Fitness Value Using Different Variations of GA

Run	1Pt-TS	2Pt-TS	1Pt-RW	2Pt-RW
1	0.999999806510745	**0.999999837454198**	1.00032908266803	0.996104323690639
2	**0.999999779249555**	0.890472899056239	0.999226396503653	0.992231041383609
3	0.99999985834495	**1.00000018720808**	1.00004471815745	1.02991416080921
4	1.09539759448256	**1.00000016438928**	1.0040721966017	1.00209612000553
5	**0.999999961895125**	0.937118637588532	0.882926664230796	1.08107453793957
6	0.99999986788425	**1.00000008290602**	1.00353489432153	0.953600464535172
7	**0.999999379549193**	1.05120935751552	1.0004624064218	1.00169884060252
8	0.99999991719546	**1.00000000096774**	1.05656052925765	1.03965111971644
9	0.999999934381845	**1.00000000487106**	0.997622958460941	0.965682109175845
10	1.03338930429439	**0.999999947112043**	1.00091621490759	0.995811022482634
11	1.11118234798637	**1.00000009134842**	0.996923355443292	1.00034180355363
12	0.999999924859209	**1.00000004938622**	1.00264070853512	0.998169628917651
13	**1.00000007145526**	0.875512548578209	0.994485090346597	1.00051609262964
14	0.836138809082159	**0.999999648336575**	0.998272650384506	1.04009335691132
15	**0.999999922907199**	1.0081279380339	0.992495747768195	0.997054754343672
16	1.01221737286364	**1.00000052776506**	0.999463931106337	1.0000982627071
17	1.04023948199726	**0.999999847582843**	1.03273179378408	0.998087248137029
18	0.851019672239198	1.02969607443561	**0.999236645921519**	0.995463082340804
19	**1.00000005473471**	1.00000007254995	1.01970992956133	0.999126063790187
20	**1.00000000641114**	1.03071506311448	0.635797572912528	1.00360265730449
21	**1.00000027044179**	1.05825445391926	0.99436677365174	0.947253159019474
22	0.97849770294463	**0.999999940769817**	0.999707508335711	1.10414983756042
23	**1.00000005974326**	0.999999793081454	1.00229797816938	1.00247306544208
24	**1.00000000303155**	1.00000009997905	1.07589413291593	1.02835649645203
25	1.00420847746269	**1.00000018413562**	1.000075574295	0.997477986775364
26	0.999999950273968	**1.00000030339609**	0.996713920675307	0.958418379169116
27	**0.999999930640142**	1.00000028028068	0.999275215395239	1.03433445656667
28	1.05253814017994	**1.00000005515217**	0.994714274375705	0.999997231342489
29	0.891780088958301	1.03026731660997	0.999370022989436	**0.999926012839511**
30	0.962989534397473	1.04955633260654	**0.998881460352044**	1.0036917944111

1Pt: One point crossover – 2Pt: Two-point crossover – TS: Tournament Selection – RW: Roulette Wheel Selection

From the results shown in table 2, the highest fitness values were when using the tournament selection along with two-point crossover. Fig.3 and Fig.4 summarize the results obtained from table 3. It is clear that tournament selection with two-point crossover gives the highest percentage (53%) of pleasing music.

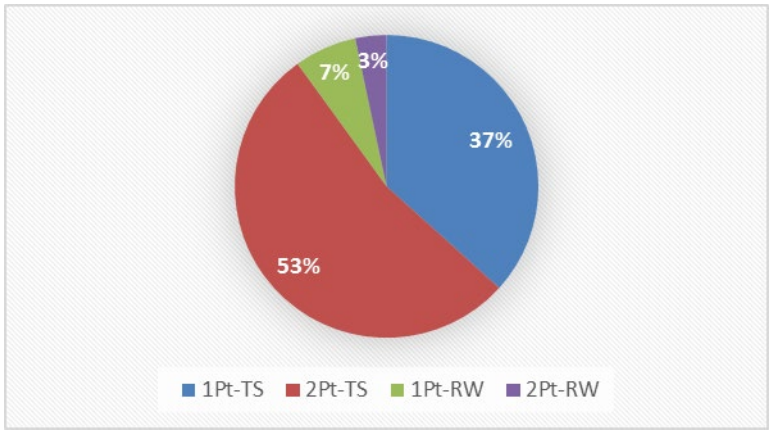

Fig.3: Highest Fitness Value Obtained From GA Variations.

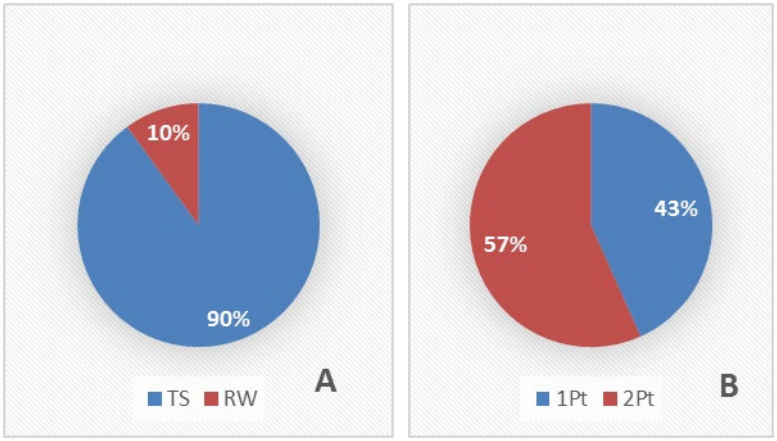

Fig.4: **A**: Highest Fitness Value Obtained Using TS and RW, and **B**: Highest Fitness Value Obtained Using 1Pt and 2Pt

4.2 Music Experts Evaluation Of Generated Music

In this evaluation, we prepared a questionnaire and asked 10 experts in the music domain from the music department in Yarmouk University to evaluate the quality of the generated musical sentences. The following are the set of questions; the music experts select their answers from a 5 likert scale:

- How do you rate the quality of the music?
- Do you think that this melody can be used as a basis to generate a new pleasing melody?
- Do you think that the music is a copy of a piece of music you know?
- Does the melody resemble the sound of one of the composer's style you know?

We selected a set of four musical patterns that is generated from the proposed algorithm using the setting on table 3. Two of them were rated high from the ANN and the other two were rated low. The error rate of the trained ANN was 0.01. We did not tell the experts about the quality of the music patterns.

Table 3: GA Setup

Number	Operator/ Parameter	Value
1	Crossover Rate	0.8 [12]
2	Mutation Rate	0.1 [12]
3	Population Size	2000
4	Generations	10000
5	Chromosome size	40 notes
6	Selection	Tournament selection
7	Crossover	Two-point crossover

In the first experiment, we evaluated the following music pattern:

m1:d=4,o=5,b=160:4b5,4b5,4p,4p,8p5,8f#5,8g5,4b5,8g5,8f5,4e5,4e5,4p,4p,4p,4p,8f5,8g5,4b,4b,4p,8p5,8f5,8g5,4b 5,8g5,4f5,4e5,4e5,4p,8p5,8f5,8g5,4b5,8g5,8f5,8e5,4p,4p,4p

This pattern was rated high by the ANN as its fitness was: 1.00008104417862. The average rating for the human evaluation to its goodness was (8.6). In the second experiment, we evaluated the following music pattern:

m2:d=4,o=5,b=250:4p,4p,16p,4p,1g5,8f5,4g5,2b5,2c5,4c5,6c#,1d#5,1e5,2c#5,1g5,8f5,4g5,2b5,2c5,4c5,6c#5,1d#5, 1e5,1c#5,32p,8p,4p,1g5,8f5,4g5,2b5,2c5,4c5,6c5,2d5,2p,4p,4p,4p

This pattern was rated highly by the ANN as its fitness was: 0.98938104417862. The average rating for the human evaluation to its goodness was (8). In the third experiment, we evaluated the following music pattern:

m3:d=4,o=5,b=250:2d#5,1g#5,1c#5,1d5,2c#5,1d#5,8a5,1p5,4g5,1b5,4f#5,2c#5,2p5,2a5,2c#5,2c5,16b5,1b5,1a#5,2 b5,8c5,2p4,2b5,1d#5,2e5,2p5,32e5,4b5,6d5,2g6,4d#7

This pattern was rated low by the ANN as its fitness was: 0.34738104017899. The average rating for the human evaluation to its goodness was (6.2). In the fourth experiment, we evaluated the following music pattern:

m4:d=4,o=5,b=250:1g#5,2c5,4c5,1g5,2c5,1g#,1f5,4f5,4a5,4c5,6d,4g#5,16f#,32f,1d#5,1a#,6a,4c5,1g5,1a#,16c,1c#, 32a,32c,1c5,6d#,8e5,16a5,4g5,2c5,32d#,2c,16b,4c,6f,16b,16d#,2f,1b,6d5

This pattern was rated low by the ANN as its fitness was: 0.47348920054623. The average rating for the human evaluation to its goodness was (6.4). From the previous four experiments we can notice that the evaluation obtained from the experts is close to the evaluation of the ANN. Fig.5 shows a comparison between the evaluation of the ANN and the human experts for the goodness of the resulted music patterns. It is evident that ANN was able to give a good indication about the quality of the generated music.

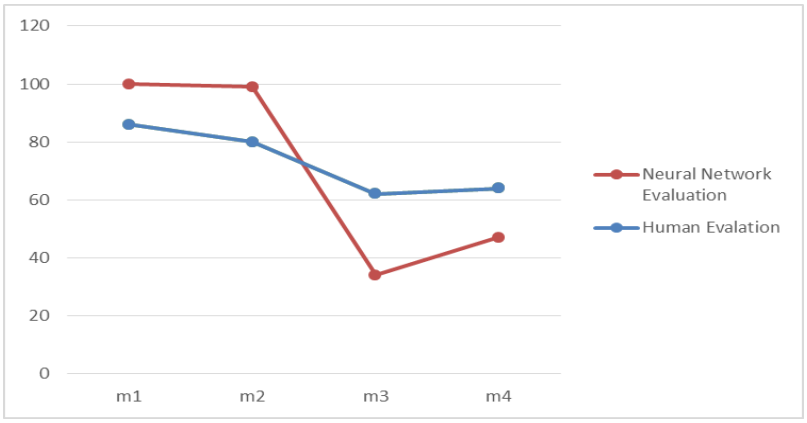

Fig.5: Human Evaluation and Neural Network Evaluation for the Goodness of the 4 Music Patterns

4.3 Accuracy and error rate of the neural network

We evaluated the accuracy and error rate of the fitness evaluation of ANN compared to the results from the human experts in the previous section. We computed the true positive rate, the true negative rate, the false positive rate and the false negative rate. Which are computed according to the values obtained from the four music patterns. The obtained results are presented in the following table:

Table 4: Confusion Matrix for 4 Music Patterns

Performance matrix	Melody
TP	2
TN	1
FP	0
FN	1

From table 4 we computed the following performance metrics [32]:

$$TP_{rate} = \frac{TP}{TP+FN} = \frac{2}{2+1} = 0.6667 \quad (1)$$

$$TN_{rate} = \frac{TN}{TN+FP} = \frac{1}{1+0} = 1 \quad (2)$$

$$FP_{rate} = \frac{FP}{TN+FP} = \frac{0}{1+0} = 0 \quad (3)$$

$$FN_{rate} = \frac{FN}{TP+FN} = \frac{1}{2+1} = 0.3333 \quad (4)$$

From these four equations, we computed the error rate and the accuracy of the ANN [32]:

$$\text{Error rate} = \frac{FP+FN}{TP+FN+FP+TN} = \frac{0+0.333}{0.6667+1+0+0.333} = 0.1665$$

$$\text{Accuracy} = \frac{TP+TN}{TP+FN+FP+TN} = \frac{0.6667+1}{0.6667+1+0+0.333} = 0.8335$$

Form the results of these equations, we found that the ANN is 83% accurate and consistent with human evaluation results.

5.0 CONCLUSION AND FUTURE WORK

The task of music composition requires the study of music concepts and investigating the rules explained by the music theory. The automatic music composition using GA and ANN is a promising approach. The approach is capable of composing different types of music, by changing the training melody type.

The type of selection in GA affects the quality of the generated music patterns. The experiments show that by using the tournament selection, we are able to generate 90% of the music patterns with high fitness values. The type of the crossover in GA affects the quality of the generated music patterns. The experiments show that by using the two-point crossover, generates 57% of the music patterns with high fitness values.

The human experts' evaluation agreed with the evaluation obtained from ANN for both good and bad patterns. The ANN was capable to evaluate the generated music patterns in an efficient way. The ANN accuracy was 83% and the error rate was 16.7%.

The future direction of this research will be to generate melodies with larger length and to generate melodies from different genres. The use of lyrics to generate the corresponding music melodies can be another direction of this research.

REFERENCES

[1] G. Salas et al., "Automatic music composition with simple probabilistic generative grammars", *Polibits*, 2011, Vol. 44, No. 44, pp. 59-65.

[2] F. Rodríguez et al., "AI Methods in Algorithmic Composition: A Comprehensive Survey", *Journal of Artificial Intelligence Research,* 2013, Vol. 48, No. 1, pp. 513-582.

[3] M. Simoni, "Algorithmic composition: a gentle introduction to music composition using common LISP and common music", *Ann Arbor, Michigan: The Scholarly Publishing Office,* 2003, the University of Michigan, University Library.

[4] R. Yagiz Mungan, "Algorithmic Composition with Virtual Instrument in MATLAB and on FPGA". *Chalmers University of Technology*, 2010, Göteborg, Sweden.

[5] D. Matić, "A genetic algorithm for composing music", *Yugoslav Journal of Operations Research*, 2010, Vol. 20, No. 1, pp. 157-177.

[6] J. Harnum, *Welcome to Basic Music Theory. In Basic Music Theory: How to Read, Write, and Understand Written Music*, 2001, ed. 2, pp. 1-11. Sol-Ut Press. ISBN: 0-9707512-9-X.

[7] M. Pilhofer and H. Day, *Music Theory for Dummies*. ed. 2. Wiley Publishing Inc. ISBN: 978-1-118-09550-8, 2011.

[8] D. Rath, "Note Identification – Music Theory". *http://donrathjr.com/note-identification-music-theory-part-11/.* Last access: Dec. 30, 2015.

[9] A. Santos et al., "Evolutionary computation systems for musical composition". In *International Conference Acoustic and Music: Theory and Applications (AMTA 2000), Iasi, Romania*. 2000, Vol. 1, pp. 97-102.

[10] M. Edwards, "Algorithmic composition: computational thinking", in *Music Communications of the ACM*, 2011, Vol. 54, No. 7, pp. 58-67.

[11] Lo, M. Y., and Lucas, S. M. "Evolving musical sequences with N-Gram based trainable fitness functions", In *Evolutionary Computation (CEC 2006), Vancouver, Canada,*. IEEE Congress, 2006, pp. 601-608.

[12] Oliwa, T., & Wagner, M. "Composing music with neural networks and probabilistic finite-state machines", In *Applications of Evolutionary Computing, Naples, Italy,* 2008, pp. 503-508.

[13] K. Cleland et al., "THE MUSIC OF CSIRAC, SOME UNTOLD STORIES", *Proceedings of the 19th International Symposium on Electronic Art (ISEA2013), Sydney*, 2013, pp. 102-110.

[14] T. H. Park, "An interview with max Mathews", *Computer Music Journal*, 2009, Vol. 33, No. 3, pp. 9-22. Massachusetts Institute of Technology.

[15] de Mantaras, R. L., and Arcos, J. L. "AI and music: From composition to expressive performance", *AI magazine*, 2002, Vol. *23, No.* 3, pp.43.

[16] J. Biles, "GenJam: A Genetic Algorithm for Generating Jazz Solos", *ICMC Proceedings, San Francisco*, 1994, pp. 131-137.

[17] Khalifa, Y and Al-Mourad, M., "Autonomous evolutionary music composer", In *Proceedings of the 8th annual conference on Genetic and evolutionary computation (GECCO '06), New York, USA*, ACM, 2006, pp. 1873-1874.

[18] B. Manaris, et al., "A corpus-based hybrid approach to music analysis and composition", In *Proceedings of the National Conference on Artificial Intelligence, Menlo Park, CA*, 2007, Vol. 22, No. 1, pp. 839-845. Cambridge, MA; London; AAAI Press; MIT Press.

[19] Chen, C. C. J., "Automatic Music Composition using Genetic Algorithm and Neural Networks: A Constrained Evolution Approach". 2000, *Department of Computer Sciences, the University of Texas at Austin*.

[20] Á. Sánchez et al., "Spieldose: An Interactive Genetic Software for Assisting to Music Composition Tasks", In *Bio-inspired Modeling of Cognitive Tasks, La Manga del Mar Menor, Spain,* 2007, pp. 617-626. Springer Berlin Heidelberg.

[21] Sheikholharam, P., and Teshnehlab, M., "Music Composition Using Combination of Genetic Algorithms and Kohonen Grammar", In *International Symposium Computational Intelligence and Design (ISCID'08), China*, 2008*,* pp. 255-260.

[22] Eck, D., and Schmidhuber, J., "A first look at music composition using lstm recurrent neural networks". 2002, *Istituto Dalle Molle Di Studi Sull Intelligenza Artificiale*.

[23] Hörnel, D., and Ragg, T., "Learning musical structure and style by recognition, prediction and evolution", In *Proceedings of the International Computer Music Conference, Greece*, 1996, pp. 59-62.

[24] D. Morris et al., "Exposing Parameters of a Trained Dynamic Model for Interactive Music Creation". In *23rd national conference on Artificial intelligence (AAAI), Chicago, Illinois,* 2008, pp. 784-791.

[25] K. Monteith, et al., "Automatic generation of melodic accompaniments for lyrics", In *Proceedings of the International Conference on Computational Creativity, Dublin, Ireland*, 2012, pp. 87-94.

[26] M. Chan et al., "Improving algorithmic music composition with machine learning". In *9th international Conference on Music Perception and Cognition (ICMPC), Italy*, 2006, pp. 73- 88.

[27] "ActiveXpertsSoftware, innovation in communications, RTTTL Format". Can be found at: http://www.activexperts.com/xmstoolkit/sms/rtttl/. Last access: Dec, 23rd, 2015.

[28] "Encog Machine Learning Framework". Can be found at**:** http://www.heatonresearch.com/encog**.** . Last access: December, 2nd, 2015.

[29] Z. G. Che et al., "Feed-forward neural networks training: A comparison between genetic algorithm and back-propagation learning algorithm", *International Journal of Innovative Computing, Information and Control,* 2011, Vol. 7, No. 10, pp. 5839-5851.

[30] M. Al-Betar et al., "An analysis of selection methods in memory consideration for harmony search", *Applied Mathematics and Computation*, 2013, Vol. 219, No. 22, pp. 10753-10767.

[31] "QuickRingTone". Can be found at: http://quick-ringtone.lightthoughts-software-inc.softalizer.com. Last access: Dec, 23rd, 2015.

[32] S. García et al., "Enhancing the effectiveness and interpretability of decision tree and rule induction classifiers with evolutionary training set selection over imbalanced problems"*, Applied Soft Computing*, 2009, Vol. 9, No. 4, pp. 1304-1314.

[33] Enrique Muñoz, Yew Soon Ong. "Memetic Music Composition". *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION*, 2016, Vol. 20, No. 1, pp. 1 - 15.

[34] Chuan-Kang Ting, Chia-Lin Wu, and Chien-Hung Liu. "A Novel Automatic Composition System Using Evolutionary Algorithm and Phrase Imitation". IEEE SYSTEMS JOURNAL, 2015, Vol. PP, No. 99, pp. 1 – 12.

[35] Sarah Opolka, Philipp Obermeier, and Torsten Schaub. "Automatic Genre-Dependent Composition using Answer Set Programming". *Twenty-first International Symposium on Electronic Art (ISEA'15)*, *Canada*, 2015, pp. 35-46.

[36] Li Su and Yi-Hsuan Yang. "Escaping from the Abyss of Manual Annotation: New Methodology of Building Polyphonic Datasets for Automatic Music Transcription". *International Symposium on Computer Music Multidisciplinary Research CMMR 2015: Music, Mind, and Embodiment, Plymouth, UK,* 2015, pp 309-321.

[37] Alsmadi, I., Alkhateeb, F., Maghayreh, E. A., Samarah, S., & Doush, I. A. (2010). Effective generation of test cases using genetic algorithms and optimization theory. Journal of Communication and Computer, 7(11), 72-82.

[38] Al Maghayreh, E., Doush, I. A., & Alkhateeb, F. (2013). Detecting distributed predicates using genetic algorithms. International Journal of Intelligent Information Technologies (IJIIT), 9(1), 56-70.

[39] Abu Doush, I., & Al-Saleh, M. I. (2017). Can genetic algorithms help virus writers reshape their creations and avoid detection?. Journal of Experimental & Theoretical Artificial Intelligence, 29(6), 1297-1310.