



Monophonic music composition using genetic algorithm and Bresenham's line algorithm

Shipra Shukla¹ · Haider Banka¹

Received: 5 August 2020 / Revised: 30 April 2021 / Accepted: 10 January 2022 /
Published online: 29 March 2022

© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2022

Abstract

Music composition is one of the oldest artistic pursuits. The role of machine in automatic generation of creative artworks, like music, is still an explorable area. In this paper, a new approach for music composition is proposed that differs from previous methods of generating music with predefined musical parameters. The proposed method gives flexibility to change the musical parameters, namely time signature, range and scale. Moreover, the user can also input any desired motif (a short piece of melody) and generate a melody that includes the motif. The type of music composed in this work is monophonic, which includes melody and rhythm. To create a proper sequence of notes, we used genetic algorithm with suitably formulated crossover and mutation operators. The rhythm is generated using Bresenham's line drawing algorithm, which has been modified to adapt different time signature with changing beats. PySynth, a simple music synthesizer, is chosen to convert music into wave file format. In the end, a comparative analysis is conducted to show the efficacy of the proposed model. Results show that the proposed algorithm performs better than conventional genetic algorithm.

Keywords Computer music · Genetic algorithm · Bresenham's line algorithm · Melody · Rhythm · Music composition

1 Introduction

Music is a creative way of conveying our emotions using medium of sound, coordinated with time. It is an organization of rhythm, melody and harmony. Music can have different textures, namely monophonic, homophonic and polyphonic. When organized with single melody and rhythm, the music has a monophonic texture. While homophonic and polyphonic textures also include harmony along with melody and rhythm. We have considered melodic and rhythmic aspect of music in this work, thereby creating monophonic music. Melody is a sequence of notes grouped as a single entity on a musical framework. Whereas

✉ Shipra Shukla
shiprashukla.iit@gmail.com

¹ Department of Computer Science & Engineering, IIT (ISM) Dhanbad, Dhanbad, India

rhythm, an essential and inescapable element of music, is a regular pattern of sound formed by combining notes of different duration. Composition of pleasing melodies and rhythms is a kind of creativity that can not be achieved by randomly walking in the search space [7]. People spend years developing the talent to compose music. Researchers have proposed several algorithms for automatic music composition to make this task easier for novice musicians [14, 36]. But most of the methods produce curbed music. In some cases, constraints are put on musical parameters to decrease the search space or to reduce complexity, which narrows down the musical possibilities. Also, the existing approaches have made fewer efforts to generate algorithmic music that integrates the users' idea. For example, Jeong et al. [21] use NSGA-II (Non-dominated Sorting Genetic Algorithm-II) to generate a variety of melodies at once. They handle the trade-off relation between stability and tension in the music. However, they use only diatonic scales for melody. They exhibit two ways for generating rhythms: the user can either select from existing material or enter the whole rhythm by himself. There is no middle way to merge the user's idea with automatic music. In this paper, we attempt to bridge this gap by providing compositional flexibility to the user. The aim of this paper is two fold: generating a method to produce a variety of compositions and merging the user's idea with algorithmic music. The proposed method offers flexibility to the user to change the musical parameters (time signature, range and scale) and insert any motif (a short piece of music) in the composition. We have utilized genetic algorithm in our work to generate melodies. The motivation for adopting GA for melody generation is that it can maintain a balance between creative exploration and continual improvement [17]. The exploration of the search space is much needed in the case of music composition to introduce diversity in music. Genetic algorithm works very effectively if appropriate fitness function and operators are used [11]. The proposed work uses genetic algorithm with domain-specific operators to produce a proper sequence of notes.

Further, we have also implemented rhythm generation, an essential component of music. In the paper [39], Toussaint et al. discussed the applicability of various algorithms from different disciplines to generate a large family of recurring rhythms (Ostinato). One such algorithm mentioned in his paper is the Bresenham's line drawing algorithm [4], which generates a similar set of rhythmic patterns. This algorithm uses only integer subtraction, addition and multiplication, which makes it simple and easy to implement. In this work, we have utilized Bresenham's line drawing algorithm to generate the rhythmic patterns for different time signatures. The purpose of automatic music generation in this work is to help novice musicians and to produce music with less effort. The main contribution of this work is mentioned as follows:

- **Computational flexibility:** The proposed system is flexible enough to work with a variety of time signatures, ranges and scales. The system also has a provision to insert any preferred motif in the composition to generate desired music.
- **Domain-specific operators:** To accomplish the objective, problem specific crossover and mutation operators are defined. Music theory rules are utilized for defining fitness function and genetic operators.
- **Time signature based rhythm:** Rhythm is generated using Bresenham's line drawing algorithm, which is formulated to adapt different time signature with changing beats.

The remaining portion of this article is arranged as follows. Next section describes the preliminaries related to our method. Section 3 reviews the previous work done in music composition domain. The details of our proposed approach is given in Section 4. Experimental results are provided in Section 5, and concluding remarks are stated in Section 6.

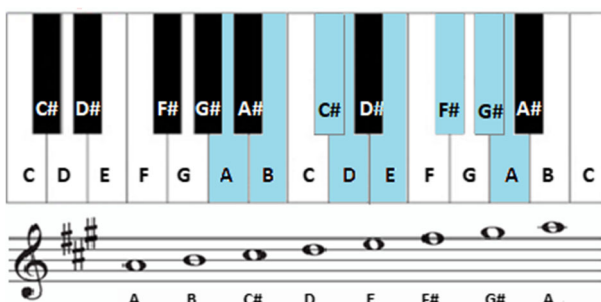
Fig. 1 3/4 time signature

2 Preliminaries

Music composition is the art of combining musical elements to create a piece of music, either vocal or instrumental. Composers often adopt the approach of extending a short piece of phrase (motif) into a considerable portion of composition [20]. This approach is also utilized in this work by adding user's motif in the melody and expending it with the help of motif-based mutation operator (discussed in Section 4.2.6). The composition generated in such a manner incorporates the user's notion with automatically generated note sequence. The essential musical terms and algorithms used in this work are briefly mentioned below. The book [19] by Hewitt M. can be referred for basic music theory rules.

2.1 Music terminologies

- **Time Signature:** It is a musical notation to determines the number of notes in each measure (bar) and type of notes. The time signature contains two numbers appearing as a fraction at the beginning of music staff, as shown in Fig. 1. The number at the top indicates how many notes are there in each bar. The bottom number indicates the type of note, namely quarter, eighth or sixteenth etc.
- **Scale:** Scale, in music theory, is an organised set of notes within an octave. There are total twelve notes available in an octave. Scales are the unique combination of notes selected out of these twelve notes as shown in Fig. 2. For instance, the chromatic scale in western music contains all twelve notes while diatonic scale contains seven and pentatonic scale contains five different notes. We could almost write an encyclopedia on different kinds of music scales from around the globe. Every scale can convey a different emotion; therefore, users can input scale as per their taste.
- **Range:** The range of melody denotes the span between the lowest note to the highest note present in that melody. Since the individuals can have different vocal ranges, this work provides an option to set the range within the comfort zone. For example, the range shown in Fig. 3 denotes that the notes of melody should be within octave 3 to 5.

**Fig. 2** A-major scale notes with sheet representation

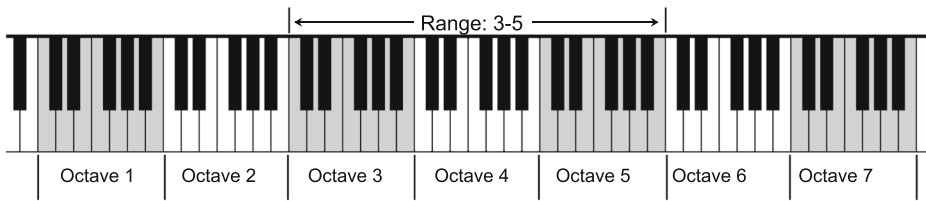


Fig. 3 Example of range and octave in piano

2.2 Genetic algorithm

Genetic algorithm (GA) is a search and optimisation method inspired by Darwin's theory of natural evolution. The process starts with a randomly generated population of individuals. In each iteration, the fitness of individuals is evaluated and more fit individuals are selected for next generation. Each individual undergoes recombination and mutation to form a new generation. In this way, we continue to evolve better individuals over generations till we reach a stopping criterion.

GA can be used for such problems where the search space is very large and the solutions can be represented by strings. This makes it a good choice for music composition where the individuals of population can be represented by note sequences. Moreover, GA can also maintain a balance between creative exploration and continual improvement, which is essential to introduce variety in music. In the paper [17], Gartland et al. described the advantages of genetic algorithm for music composition, which also inspired us to use GA in our work.

2.3 Bresenham's line algorithm

Bresenham's line algorithm is used to draw a line in raster graphics. The raster graphics contain bitmaps which are 2-D grids of pixels. In the algorithm, the pixels of raster are coloured depending on the slope of the line in order to draw a close estimate of straight line. If the coordinates of endpoints (x_1, y_1) and (x_2, y_2) are given, the algorithm finds all intermediate coordinates required to draw the line. To calculate the coordinates of line (having slope < 1), the value of x is increased by 1 in every step and the value of y is chosen, whether it is y or $(y + 1)$. A decision parameter d is used to select the y coordinate for next point. If the decision parameter $d < 0$ is true, then y remains same otherwise the value of y is incremented by 1. The steps involved in Bresenham's line algorithm is described in Algorithm 1.

We have used this algorithm in our work to create recurring rhythmic patterns. Bresenham's line drawing algorithm is one of the wide variety of algorithms mentioned by Toussaint et al. [39] to generate recurring patterns. The idea behind generating rhythm is to spread out the onset of notes as evenly as possible across the available beats. This problem is similar to drawing a straight line in raster graphics, in which the rises of the y -coordinate must be spread out as uniformly as possible over the x -coordinate. This describes the suitability of using Bresenham's line algorithm in our work to create rhythms. The algorithm is also formulated to adapt different time signature and beats. The rises in the value of y is denoted by '1' while no change is denoted by '0'. The binary sequence generated by algorithm represents the rhythm pattern where a '0' bit represents the continuation of previous note and '1' bit represents note onset.

Algorithm 1 Bresenham's Line Algorithm (for slope < 1).

Input : Start point (x_1, y_1) , End point (x_2, y_2)

- 1 Enter the value of (x_1, y_1) , (x_2, y_2)
- 2 Consider (x, y) as starting point, $x = x_1, y = y_1$
- 3 Calculate: $dx = x_2 - x_1, dy = y_2 - y_1$
- 4 Calculate: $d = 2*dy - dx$
- 5 Check if $x \geq x_2$, then Stop.
- 6 Draw the co-ordinates (x, y)
- 7 If $d < 0$ then $d = d + 2dy$
- 8 If $d \geq 0$ then $d = d + 2dy - 2dx, y = y + 1$
- 9 Increment $x = x + 1$
- 10 Go to step 5

Output: Intermediate coordinates

3 Literature review

In recent years, algorithmic music composition has become very popular among computer researchers [16]. Many researchers have devoted their effort to generate automatic music or its components using different approaches. The methods that achieved considerable success in composing automatic music or parts of music includes rule-based methods [13, 25], evolutionary algorithm [26], machine learning [15] neural networks [27], fuzzy logic [40], grammatical evolution [12], artificial immune system [30, 31], relation-based approach [6], probabilistic model [34] and reinforcement learning [33] etc. We have listed some considerable works for music composition in Table 1. In the studies, we note that the meta-heuristic approaches have been widely used in the field of music composition [1, 22, 35]. One of the well-known approaches is genetic algorithm which has given effective results in this domain [29]. Based on the evaluation technique used, attempts using GA can be categorized as interactive and automatic.

In the interactive method, fitness of the music sample is evaluated by audience themselves. Biles et al. [3] designed GenJam to generated jazz solos using interactive GA. The difficulty in choosing efficient algorithmic fitness function can be eliminated by using nteractive evaluation. However, the user has to listen to each sample thoroughly to give feedback which limits the population size and number of generations. After some evaluations user can get tired and lose interest. The music produced in such a way is biased towards the evaluator's taste. Johanson et al. [23] used this approach to design GP-music, an interactive system to create short musical patterns. The system is extended by using neural network based automatic fitness rater, which learns from the users' rating data. Tokui et al. [38] used an interactive evolutionary computation to develop CONGA, a musical rhythm generator.

To evict the interactive fitness bottleneck, some researchers have replaced human evaluators with automatic methods like knowledge-based and learned fitness functions. Papadopoulos et al. [32] used algorithmic fitness function to generate jazz melodies over the given chord progression. Similarly, Matic [28] composed melodies with automatic genetic approach. Kikuchi et al. [24] used chord progression for melody generation by genetic algorithm. Ting et al. [37] have utilized human composed music as a reference to generate imitation by evolutionary algorithm.

Some other exemplary approaches include data-driven methods for automatic composition [9]. Colombo et al. [8] used deep recurrent neural network (RNN) with gated

Table 1 Some popular approaches for music composition

Reference	Composition Approach	Remark
Biles et al. [3]	Interactive GA for melody (Jazz solos). Rhythm and other elements taken from progression file	Uses hierarchical population representation
Johanson et al. [23]	Interactive Genetic Programming for melody	Extended the system using neural network based fitness evaluator
Tokui et al. [38]	Interactive evolutionary algorithm for rhythm generation	Combines GA and genetic programming
Papadopoulos et al. [32]	GA for Jazz melody and rhythm	Uses input chord progression for reference
Matic [28]	GA for melody and rhythm	Uses reference individual to initialize rhythm and other musical parameters
Kikuchi et al. [24]	GA for melody	Features of rhythm, pitch, and chord progression are extracted from sample melody
Ting et al. [37]	GA for melody. Rhythm, scale and chord taken from sample music	Uses sample melody for phase imitation.
Jeong et al. [21]	Multi objective evolutionary approach for melody. Rhythm determined by user	Musical stability and tension are used as fitness measures
Colombo et al. [8]	Deep learning for monophonic music	Corpus of Irish folk songs is used to generate music with same style
Guedes et al. [18]	Carnatic Rhythm using data-driven methods	Uses corpus of carnatic composition in adi taal
Agrawala et al. [2]	Generative RNN model for sheet music	Uses dataset in ABC music notation

recurrent unit (GRU) to generate convincing monophonic melodies. Guedes et al. [18] used data-driven method to generate carnatic rhythm using the corpus of carnatic percussion composition. Agrawala et al. [2] proposed generative RNN models to build a music generator using Seq2Seq and Character RNN, which is trained by Abc formatted music dataset with approximately 34,000 songs of different genres. In such data based methods, we have to train the model with large amount of data each time the user's choice changes which is not always possible. These methods generate music that is akin to the music in the dataset. While the proposed algorithm strives to come up with new melodies. The music composed by the proposed work also incorporates the user's motif to create desired music, which distinguishes it from the existing approaches.

4 Proposed method

The proposed model can work in two modes, i.e., default and manual, as shown in Fig. 4. In default mode, all the musical parameters (time signature, range and scale) are predefined, and length of user's motif is zero. Whereas in manual mode, the user can change any of the

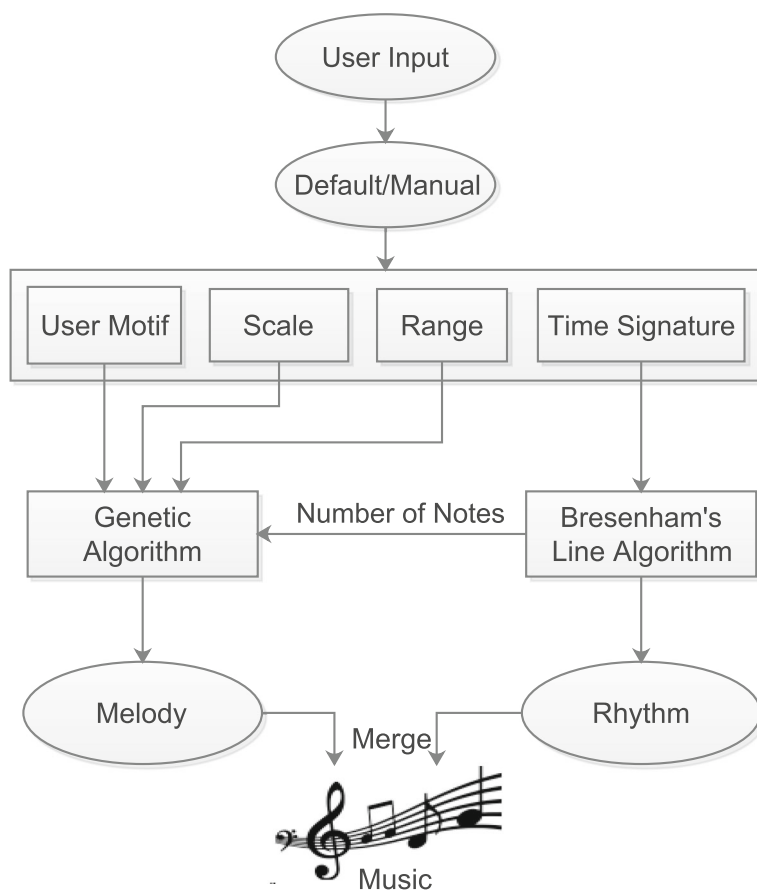


Fig. 4 Scheme of proposed model

musical parameters and also input any motif. The user's input is passed to the algorithms to generate rhythm and melody. The output rhythm and melody are merged to produce the music piece.

4.1 Rhythm generation

The time signature given by user is used to calculate beat duration and total number of beats in the rhythm. The rhythm pattern is generated for single phrase having eight bars. Figure 5 illustrates the steps involved in the process. To understand the process of rhythm generation, let us consider the case with 8 bars and 4/4 time signature. Now, we can obtain the following values.

$$\text{Beats per bar}(Bt) = \text{Top number of Time Signature} = 4$$

$$\text{Beat value}(V) = \text{Bottom number of Time Signature} = 4$$

$$\text{Number of Bars}(B) = 8$$

We calculate the total length of rhythm by taking the beat duration w.r.t. beat value from Table 2. In this example the beat value is 4 so the duration of beat is $1/4$.

$$\begin{aligned} \text{Length of rhythm} &= B * Bt * \frac{1}{V} \\ &= 8 * 4 * 1/4 \\ &= 32 * 1/4 \end{aligned}$$

Thus, thirty-two quarter beats are required to generate total length of rhythm for 8 bars with 4/4 time signature. If we replace the quarter beat with the eighth beat, we need a double number of beats to generate the same length. In general, if the beat duration $\frac{1}{V}$ is changed

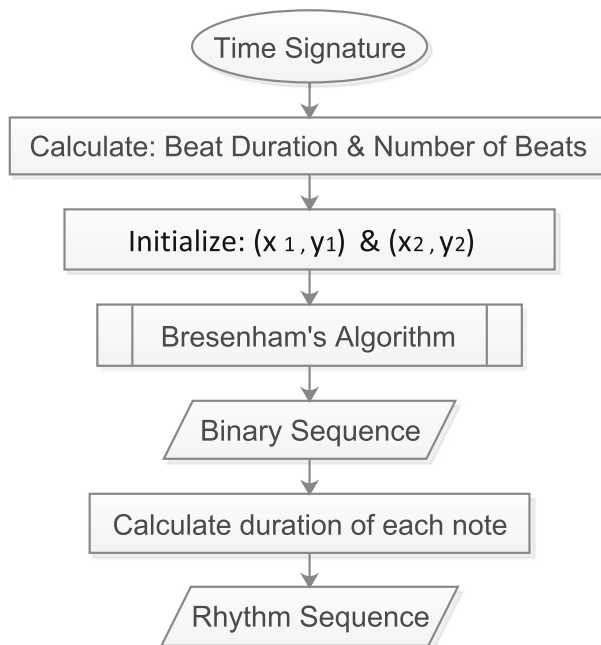







Fig. 5 Overview of rhythm generation process

Table 2 Type of Beats used in music theory

Beat Value	Name	Duration	Representation
1	Whole	1	
2	Half	1/2	
4	Quarter	1/4	
8	Eighth	1/8	
16	Sixteenth	1/16	

to $\frac{1}{2^n * V}$ then the total number of beats required will be $2^n (B * Bt)$ where n is defined as:

$$n = \left\{ x \mid x \in \mathbb{Z}, \frac{1}{16} \leq \frac{1}{2^x * V} \leq \frac{1}{2} \right\}$$

The value of n ensures that the beat duration lies within the range $[\frac{1}{16}, \frac{1}{2}]$. Although there are other durations possible but they are less popular and seldom used. When the rhythm generator receives a time signature from the user, it calculates beat duration $\frac{1}{2^n * V}$ and total number of beats $2^n (B * Bt)$ by taking random value of n . By doing so, we can generate different beat duration for the same time signature and bring variety in rhythm. The number of beats calculated above is passed in Algorithm 2 to initialize the endpoints. The slope of line is taken from 0 to 1. The endpoints of the line are initialized as follows:

$$\begin{aligned} (x_1, y_1) &= (0, 0) \\ x_2 &= 2^n (B * Bt) \\ y_2 &= [0, x_2] \end{aligned}$$

Algorithm 2 Bresenham's Line Algorithm for Rhythm.

Input : Start point (x_1, y_1) , End point (x_2, y_2)

- 1 Enter the value of (x_1, y_1) , (x_2, y_2)
- 2 Consider (x, y) as starting point, $x = x_1$, $y = y_1$
- 3 Calculate: $dx = x_2 - x_1$, $dy = y_2 - y_1$
- 4 Calculate: $d = 2 * dy - dx$
- 5 Check if $x \geq x_2$, then return S and Stop.
- 6 If $d < 0$ then $d = d + 2dy$, Append 0 to S
- 7 If $d \geq 0$ then $d = d + 2dy - 2dx$, $y = y + 1$, Append 1 to S
- 8 Increment $x = x + 1$
- 9 Go to step 5

Output: Binary Sequence (S)

The algorithm generates binary sequence, as we can see in Fig. 6. Each bit denotes the beat duration ($\frac{1}{2^n * V}$). Bit value 1 implies the beginning of a new note, while bit value 0 implies continuation of previous note. Figure 7 shows conversion of binary sequence to note duration.

In this way, we have obtained the number of notes and duration of each note. The number of notes will serve as the length of chromosome in melody generation.

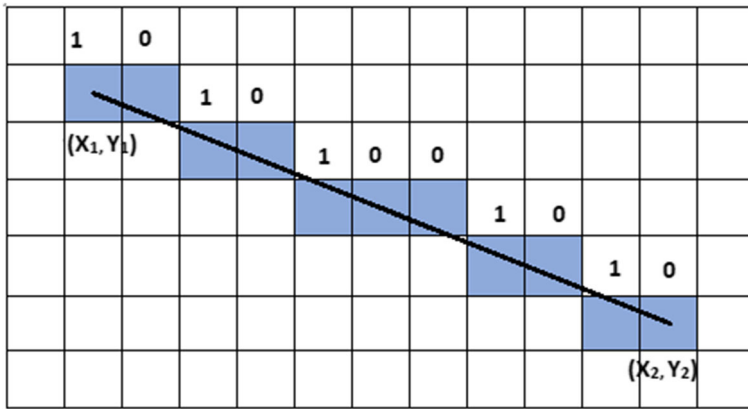


Fig. 6 Example of output binary sequence with (x_1, y_1) and (x_2, y_2) as input

4.2 Melody generation

The user given scale, range and motif are utilized in melody generation. The motif provided by user is included in the chromosomes while generating initial population. The population is evaluated based on fitness functions. If the criterion is satisfied, then output melody is generated; else, the population is passed for the next step. Further, selection, crossover and mutation operations are performed to generate new population. This cycle continues until the criterion is fulfilled. Below is an elaboration of every step included in the process.

4.2.1 Knowledge representation

Chromosome is represented as a sequence of triples. Each triple contains information as $\langle \text{Note index, Octave index, Binary bit} \rangle$. An example of chromosome is given in Fig. 8.

- Note index (NI) represents the note of the scale. Notes along with rest are encoded in decimal numbers, as illustrated in Table 3.
- Octave index (OI) represents the octave of the current note. These octaves are also encoded in decimal number based on the user given range, as shown in Table 4.
- Binary bit ‘1’ is appended to the notes belonging to user’s motif while bit ‘0’ is appended to the randomly generated notes.

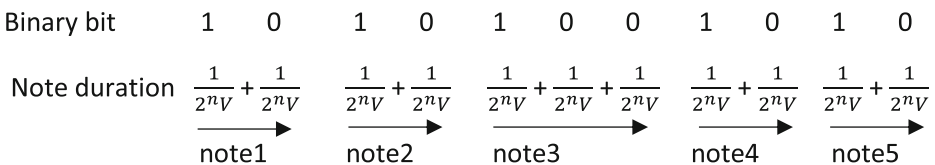


Fig. 7 Example of converting a binary sequence to note duration

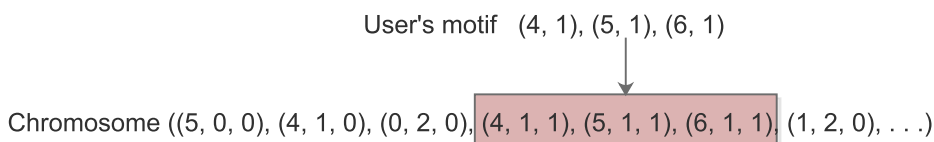


Fig. 8 Example of chromosome with user's motif

4.2.2 Initial population

The population is initialized with 50 chromosomes. Each chromosome is a combination of user given motif and randomly initialized triples as described in Fig. 8. The length of chromosome (L) denotes the number of notes in melody, which is obtained from the rhythm generation phase. The length of user motif (U) may vary, but the total length of chromosome is fixed. So, the number of randomly initialized triples (R) is

$$R = L - U$$

R number of tuples are initialized, and the user motif is attached at a random position. Now the chromosome is ready to be processed further.

4.2.3 Evaluation

The fitness function for evaluating the population is defined in Eq. (1). It is a weighted sum of four functions. A higher fitness value represents a better chromosome.

$$Fitness = w_1 * ON + w_2 * TF + w_3 * NJ + w_4 * RN \quad (1)$$

- **Octave Note Fitness (ON):** Keeping the notes within an octave makes a melody singable without strain. This function counts the number of notes found in the same octave within a chromosome. Then returns the largest number of notes found in an octave.
- **Triad Fitness (TF):** A melody that remains on the same pitch can easily get boring. Good tunes have rising and falling melodies. As the melody proceeds, the pitches can gradually or rapidly go up or down. This fitness function looks at every set of three adjacent notes and awards a point if they either all rise or fall. As we can not directly compare the notes of different octaves, so we calculate the absolute value of the note using Eq. (2). Then we use Eq.(3) to find the triad fitness of the given chromosome. A constant value 2 is added in the Eq. (3) so that the maximum triad fitness is equal to the length of chromosome.

$$Ab = OI * S + NI \quad (2)$$

$$TF = 2 + \sum_{i=1}^N f_1(Ab_i, Ab_{i+1}, Ab_{i+2}) \quad (3)$$

Table 3 Example of note index mapping

Scale Notes	c	d	e	f	g	a	b	rest
Note Index	0	1	2	3	4	5	6	7

Table 4 Example of octave index mapping

Range (3-5)	3	4	5
Octave Index	0	1	2

$$f_1(Ab_i, Ab_{i+1}, Ab_{i+2}) = \begin{cases} 1, & Ab_i < Ab_{i+1} < Ab_{i+2} \\ 1, & Ab_i > Ab_{i+1} > Ab_{i+2} \\ 0, & \text{otherwise} \end{cases}$$

where Ab_i is the absolute value of i^{th} note in the chromosome, OI is octave index, NI is note index and S is number of notes in the scale.

- **No Jump Fitness (NJ):** Singable tunes do not contain large intervals. The function defined in Eq. (4) awards a point for every jump that is within a perfect fourth. A constant value 1 is added in the Eq. (4) so that the maximum fitness is equal to the length of chromosome.

$$NJ = 1 + \sum_{i=1}^N f_2(Ab_i, Ab_{i+1}) \quad (4)$$

$$f_1(Ab_i, Ab_{i+1}) = \begin{cases} 1, & |Ab_i - Ab_{i+1}| < MaxJump \\ 0, & \text{otherwise} \end{cases}$$

- **Repeat Note Fitness (RN):** Too much repetition of a note in consecutive manner does not sound good. This fitness function penalizes the genome having consecutive note repetition. Penalty is equal to the maximum consecutive occurrence of a note within a chromosome.

4.2.4 Selection

Selection is used at two places:

- **Parent Selection:** Roulette Wheel method with Fitness Proportionate Selection is used to select the parent genomes for crossover. The method consists of choosing a probability of selection that depends on the fitness level of the genome. The genomes having higher fitness have a higher probability of selection. If the fitness of i^{th} genome is F_i , the probability of its selection is defined as:

$$p_i = \frac{F_i}{\sum_{i=0}^N F_i} \quad (5)$$

where N denotes number of genomes in the population.

- **Survivor Selection:** The $(\mu + \lambda)$ -Evolutionary Strategy is used for survivor selection. Costa et al. [10] has shown the effectiveness of using $(\mu + \lambda)$ -GA over classic GA. This is an elitist selection method where λ descendants are produced with μ parents, where $(\lambda > \mu)$. The descendants, along with their parents, participate in the selection of the best μ individuals to create the next generation.

In the proposed work, Roulette Wheel method is used to select parents for crossover from the pool of μ parents. This crossed population is then passed for three different

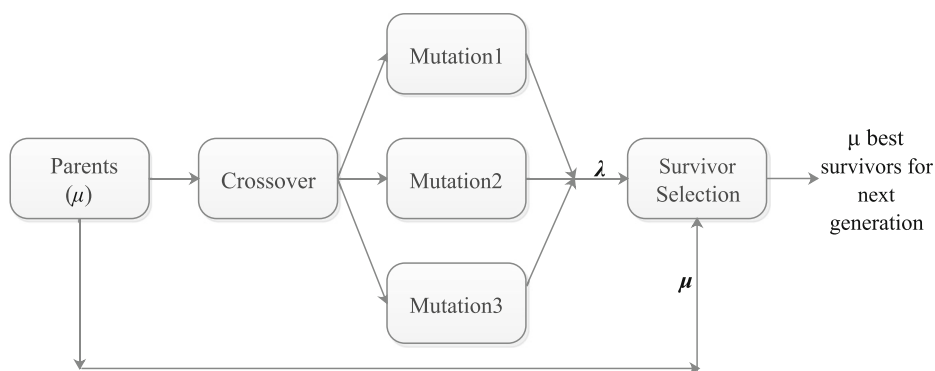


Fig. 9 $(\mu + \lambda)$ -Strategy for Survivor Selection

mutation operators as shown in Fig. 9. The output of all the three operators is combined to a pool of λ off-springs. Now the μ parents and λ off-springs participate in the selection of μ best individuals for next generation.

4.2.5 Crossover

We have opted for the single point crossover operator. As the chromosomes contain the user's motif, the crossover point should therefore be selected in such a way that it preserves the user pattern until the end. Keeping this in mind, we have created a set of possible crossover points for each selected pair of parents. Below mentioned steps are followed for picking a crossover point.

- Referring to Fig. 10, the points that lie within the user motif are eliminated from the available set of points (as shown in Eq. (6)).

$$C = \{C_1, C_2, C_3, \dots, C_{n-1}\} - \{C_i, C_{i+1}, \dots, C_j, C_{j+1}, \dots\} \quad (6)$$

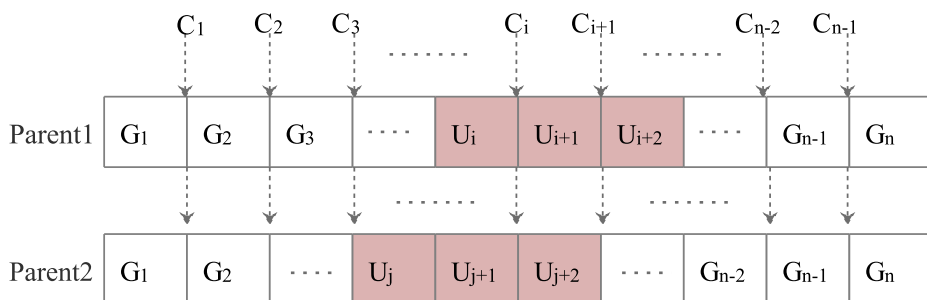


Fig. 10 Structure of parent genomes selected for crossover

where:

$U \rightarrow$	User given gene
$G \rightarrow$	GA generated gene
$C \rightarrow$	Set of possible crossover points
$i, j \rightarrow$	Starting position of user input in parent1 and parent2 respectively

- Now, for each point in set C , we check whether the user motif lies in the same direction from the point in both the parents. If not, remove the point from the set.
- Then the melodic interval (d) is calculated for the remaining points in set using Eq. (7).

$$d_k = \max(|Ab1_k - Ab2_{k+1}|, |Ab2_k - Ab1_{k+1}|) \quad (7)$$

where $Ab1_k$ and $Ab2_k$ refers to the absolute value of k^{th} note in parent 1 and parent 2 respectively and d_k is the melodic interval for k^{th} point in set C . The absolute value is calculated using Eq. (2).

- Finally the point having minimum melodic interval is chosen as the crossover point for the selected pair of parents. It will avoid pitch jumps that may occur due to crossover.

4.2.6 Mutation

Mutation operators are used to maintain diversity in the population. Three mutation operators are used in this work. All the operators are defined in such a way that they preserve the user motif until the end of the process.

- **Note based mutation** replaces a randomly chosen note with such a note which is upto 4 intervals higher or lower than the neighbours of the original note. An interval greater than 4 can disrupt the melody. The motif given by user is not replaced during note based mutation.

If i^{th} note is chosen for replacement and its neighbour notes are at $(i-1)$ and $(i+1)$ position then new note (Ab_{new}) is computed as

$$Ab_{new} = \begin{cases} S_L \cap S_R, & \text{if } (S_L \cap S_R) \neq \emptyset \\ S_L \cup S_R, & \text{otherwise} \end{cases} \quad (8)$$

$$S_L = \{x | Ab_{i-1} - 4 \leq x \leq Ab_{i-1} + 4\}$$

$$S_R = \{x | Ab_{i+1} - 4 \leq x \leq Ab_{i+1} + 4\}$$

where S_L is a set of notes that lie within 4^{th} interval from the left neighbour. S_R is a set of notes that lie within 4^{th} interval from the right neighbour. Equation (8) ensures that the new note is closer to at least one of the neighbours.

- **Motif based mutation** copies a randomly picked fragment or its inversion to a different location. This repetition of notes can make the music more interesting and catchy. To avoid excessive repetition, the fragment size is restricted to 3. The user motif is also considered for picking the fragments but its notes are not replaced.
- **Shuffle mutation** selects a small fragment from chromosome and shuffles the order of notes. A small change in the note positions can increase the fitness of genome. The user motif is not considered for shuffling.

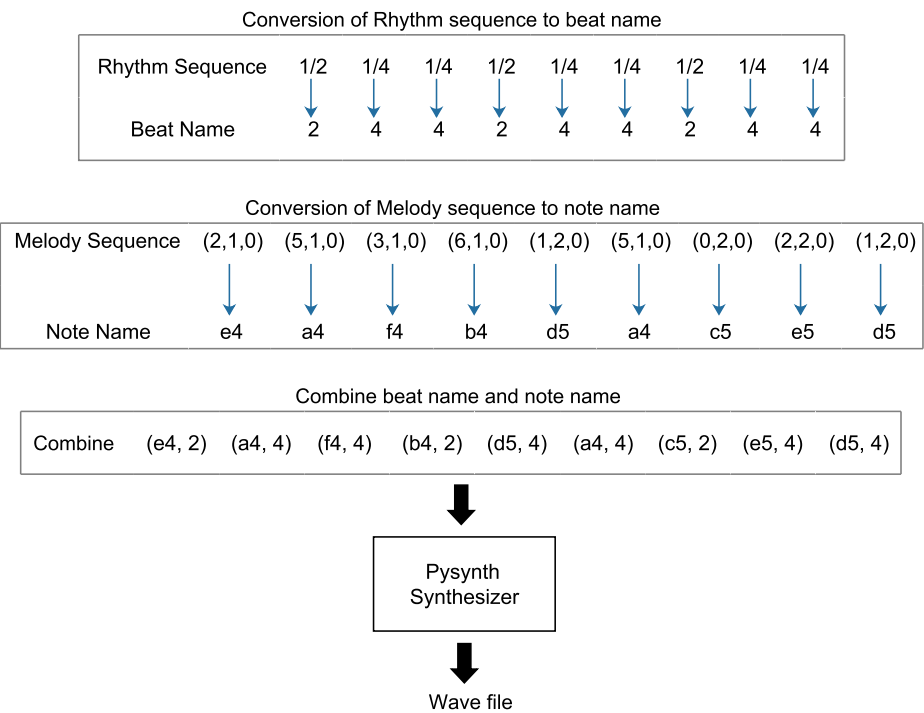


Fig. 11 Example for converting the output to wave file

5 Results and discussion

This section presents the compositions generated by the proposed approach and analysis of results. We have chosen a very well known AABA form to compose music. It consists of four sections with eight bars each, where the first two sections repeat (AA), the third section (i.e. B) is different, which creates tension in music, and the last section brings us back to the first section (A). The rhythm for a single section is generated using Algorithm 2 and the same rhythm is repeated for all four sections. The melody for section A and B is generated using the proposed Genetic algorithm. Once sections A and B are complete, we copy and combine them into a full AABA song.

The final output is generated in the wave file format using a music synthesizer named PySynth. Any rendered genome gets converted into a variation of DOT notation that’s compatible with PySynth. An example of conversion is given in Fig. 11. First, the output of rhythm generation phase is converted into beat name using Table 2 and the output of melody generation phase is converted into note names as given in Tables 3 and 4. After that, PySynth converts it to a wave file. We have uploaded some of the melodies on YouTube.¹ Further in this paper, we have shown sheet representation of the generated music for two different cases along with the parameters used.

¹<https://www.youtube.com/watch?v=BDJbspF-vA0&feature=youtu.be>



Fig. 12 Melody generated using default parameters

5.1 Case-I (default)

In this case, the melody is generated without any user given parameters and motif. The default parameters used to produce the melody are c-major scale, 4/4 time signature and 3-7 octave range. Figure 12 shows sheet representation of the generated melody.

5.2 Case-II (manual)

In this case, we have modified the musical parameters and also provided a motif ('c4', 'eb4', 'd4'). The changed musical parameters are 3/4 time signature, c-minor scale, 3-6 octave range. As we can see in Fig. 13, a 32-bar piece of music is generated. The given motif is mixed with the melody and preserved in each section which is highlighted in red box.

Due to domain-specific genetic operators, the GA converged very rapidly to high fitness. Since the fitness of genome depends on its length, the genome having different length can have different highest fitness value. When average fitness of genome reaches 90%, the algorithm terminates. Short length genomes converge very quickly as compared to long length genomes. Figures 14 and 15 illustrates the convergence of genome having length 26 and 43 respectively.



Fig. 13 Melody generated using user given parameters

To examine the performance of proposed method, we compared it with existing genetic algorithm operators. In the first test, we have compared the proposed algorithm with conventional genetic algorithm having single point crossover, random resetting mutation and roulette wheel selection. Size of the population is 50, and rest of the parameters are same in both algorithms. After 20 generations, the average fitness of the population is calculated. The test is performed 100 times, and the fitness value is plotted in a scattered plot shown in

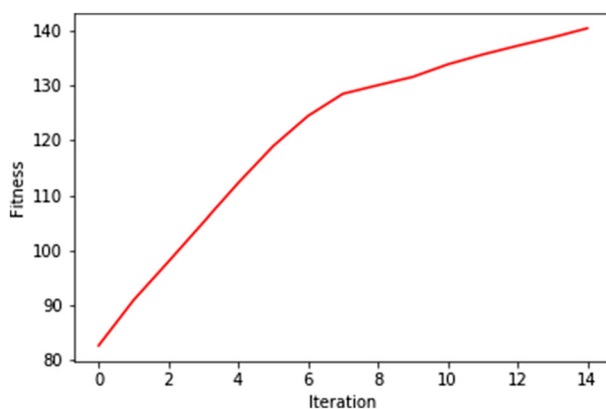


Fig. 14 Fitness vs iteration graph of genome having length 26

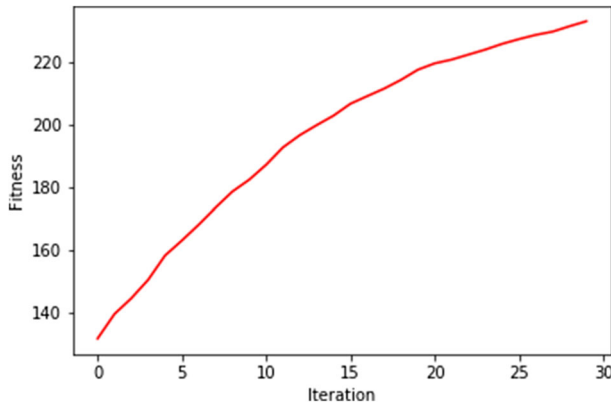


Fig. 15 Fitness vs iteration graph of genome having length 43

Fig. 16. As we can see, the proposed algorithm has performed better for the given evaluation function.

In next comparison, we have used genetic algorithm with single point crossover on one side and proposed crossover on the other. The size of population is 50, and rest of the operators and parameters are same in both algorithms. The average fitness of population is calculated after 20 generations. This test is also performed 100 times, and results are shown in Fig. 17. Tests have shown that the proposed operators are more effective than existing operators for the given fitness function.

Considering the above results and analysis, we can say that the method is capable of achieving the following outcomes.

- The proposed method is flexible to work with a variety of scales, ranges, and time signatures, as we can see in Figs. 12 and 13, two music pieces with different musical features.
- The algorithm is compared with the conventional genetic operators to examine the performance. The results are promising (refer to Figs. 16 and 17).

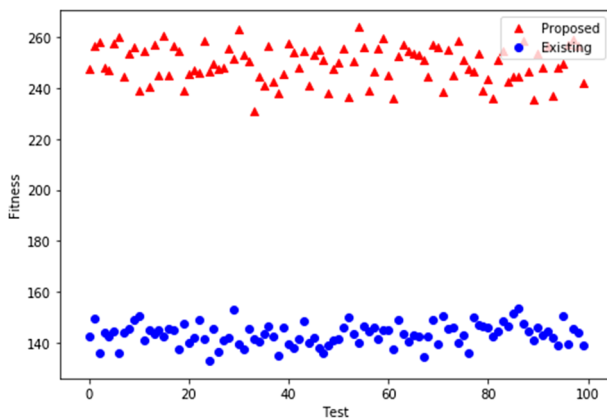


Fig. 16 Proposed algorithm vs existing algorithm

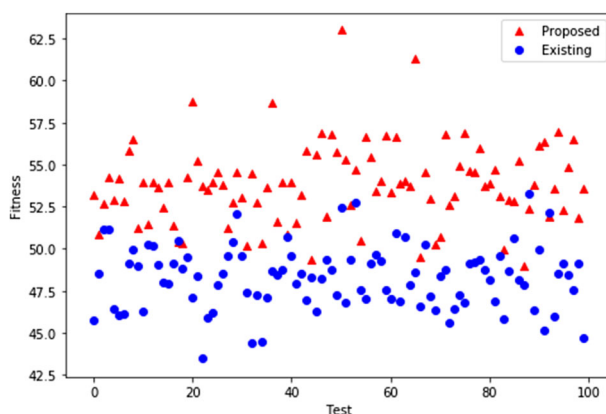


Fig. 17 Proposed crossover vs existing single point crossover

- Generates rhythm sequence for different time signatures and changing beat durations using suitably formulated Bresenham's line algorithm.
- The algorithm combines the user's motif with computer generated melody (refer to Fig. 13). To achieve this, new crossover and mutation operators are defined.

Also, We have compared some aspects of existing works in this area with the proposed method.

- **Proposed method vs existing data-driven approaches:** When a data-driven system is trained on significant number of musical artifacts, it can only perpetuate what has already been documented [5]. These algorithms perform well in style imposition tasks [2, 8, 18]. When it comes to generating new musical content, such algorithms tend to imitate the training set instead of exhibiting true creativity. While proposed method generates new compositions with creative exploration of the search space.
- **Proposed method vs existing evolutionary approaches:** The existing methods on generating monophonic music mostly work in two directions: creating compositions identical to the given reference [24, 28, 37] or generating new melodic ideas [3, 23]. When it comes to generating music similar to the reference, the proposed method is not a suitable option. This method is adaptable to generate new melodies that also incorporate user's idea, making it different from existing works.
- **Proposed rhythm generation vs existing rhythm generation approaches:** Most of the previous studies discussed in this paper use different approaches to generate a different set of rhythms [18, 38]. The proposed method for rhythm generation is inspired by the work of Toussaint et al. [39]. In the proposed work, Bresenham's line algorithm integrates two additional aspects of music, i.e. time signature and beat duration (which is not covered in [39]), to generate rhythmic patterns.

6 Conclusion and future work

In this paper, we have discussed a new approach to algorithmic music composition that generates melody based on user given musical parameters and also converges very fast. The proposed method consists of two phases, namely rhythm generation and melody generation.

The parameters of the user's choice are utilized in both phases. In the first phase, we have generated rhythm sequences using Bresenham's line algorithm. In second phase, we have generated pitch sequences using genetic algorithm with domain-specific operators. Finally, both the rhythm and pitch sequences are combined to achieve monophonic music. The readers can listen to some melodies created by the proposed method on YouTube.² The use of a domain-specific crossover operator is one of the reasons for obtaining better results. Survivor selection process and mutation operators also contribute to the enhanced performance of the proposed method. Motif based mutation includes repetition of notes which makes the melody more musical and interesting. Tests have shown that the proposed algorithm converges very fast and works more effectively than exiting GA for the given fitness function. In future, we will expand this work by also considering the harmonic aspect of music. Further, we will also study the state-of-the-art algorithms to produce more soothing and human-like music.

References

1. Acampora G, Cadenas JM, De Prisco R, Loia V, Muñoz E, Zaccagnino R (2011) A hybrid computational intelligence approach for automatic music composition. In: 2011 IEEE International conference on fuzzy systems (FUZZ-IEEE 2011). IEEE, pp 202–209
2. Agarwala N, Inoue Y, Sly A (2017) Music composition using recurrent neural networks CS 224. Natural Language Processing with Deep Learning. Spring
3. Biles JA et al (1994) Genjam: a genetic algorithm for generating jazz solos. In: ICMC, vol 94, pp 131–137
4. Bresenham JE (1965) Algorithm for computer control of a digital plotter. IBM Syst J 4(1):25–30
5. Briot JP, Pachet F (2017) Music generation by deep learning-challenges and directions. arXiv:171204371
6. Cao X, Sun L, Niu J, Wu R, Liu Y, Cai H (2015) Automatic composition of happy melodies based on relations. Multimed Tools Appl 74(21):9097–9115
7. Cohen H (2002) A self-defining game for one player: on the nature of creativity and the possibility of creative computer programs. Leonardo 35(1):59–64
8. Colombo F, Muscinelli SP, Seeholzer A, Brea J, Gerstner W (2016) Algorithmic composition of melodies with deep recurrent neural networks. arXiv:160607251
9. Cope D (1996) Experiments in musical intelligence, vol 12. AR editions
10. Costa EO, Pozo A (2006) A $(\mu + \lambda)$ -gp algorithm and its use for regression problems. In: 2006 18th IEEE International conference on tools with artificial intelligence (ICTAI'06). IEEE, pp 10–17
11. Davis L (1991) Handbook of genetic algorithms
12. de la Puente AO, Alfonso RS, Moreno MA (2002) Automatic composition of music by means of grammatical evolution. In: Proceedings of the 2002 conference on APL: array processing languages: lore, problems, and applications, pp 148–155
13. Delgado M, Fajardo W, Molina-Solana M (2009) Inmamusys: intelligent multiagent music system. Expert Syst Appl 36(3):4574–4580
14. Delgado M, Fajardo W, Molina-Solana M (2011) A state of the art on computational music performance. Exp Syst Appl 38(1):155–160
15. Eck D, Schmidhuber J (2002) A first look at music composition using lstm recurrent neural networks. Istituto Dalle Molle Di Studi Sull Intelligenza Artificiale 103:48
16. Fernández JD, Vico F (2013) Ai methods in algorithmic composition: a comprehensive survey. J Artif Intell Res 48:513–582
17. Gartland-Jones A, Copley P (2003) The suitability of genetic algorithms for musical composition. Contemp Music Rev 22(3):43–55
18. Guedes C, Trochidis K, Anantapadmanabhan A (2018) Modeling carnatic rhythm generation: a data-driven approach based on rhythmic analysis. In: Proceedings of the 15th sound & music computing conference

²<https://www.youtube.com/watch?v=BDJbspF-vA0&feature=youtu.be>

19. Hewitt M (2008) Music theory for computer musicians. Course Technology. CENGAGE Learning
20. Jarrett S, Day H (2008) Music composition for dummies. John Wiley & Sons
21. Jeong J, Kim Y, Ahn CW (2017) A multi-objective evolutionary approach to automatic melody generation. *Expert Syst Appl* 90:50–61
22. Jeong JH, Ahn CW (2015) Automatic evolutionary music composition based on multi-objective genetic algorithm. In: *Proceedings of the 18th Asia Pacific symposium on intelligent and evolutionary systems-vol 2*. Springer, pp 105–115
23. Johanson B, Poli R (1998) GP-music: an interactive genetic programming system for music generation with automated fitness raters. University of Birmingham, Cognitive Science Research Centre
24. Kikuchi M, Osana Y (2014) Automatic melody generation considering chord progression by genetic algorithm. In: *2014 Sixth World congress on nature and biologically inspired computing (NaBIC 2014)*. IEEE, pp 190–195
25. Kirke A (2019) Applying quantum hardware to non-scientific problems: Grover’s algorithm and rule-based algorithmic music composition. [arXiv:190204237](https://arxiv.org/abs/190204237)
26. Loughran R, O’Neill M (2020) Evolutionary music: applying evolutionary computation to the art of creating music. *Genet Program Evolvable Mach*, 1–31
27. Mao HH, Shin T, Cottrell G (2018) Deepj: style-specific music generation. In: *2018 IEEE 12th international conference on semantic computing (ICSC)*. IEEE, pp 377–382
28. Matic D (2010) A genetic algorithm for composing music. *Yugoslav J Oper Res* 20(1):157–177
29. Miranda ER, Al Biles J (2007) Evolutionary computer music. Springer
30. Navarro M, Caetano M, Bernardes G, de Castro LN, Corchado JM (2015) Automatic generation of chord progressions with an artificial immune system. In: *International conference on evolutionary and biologically inspired music and art*. Springer, pp 175–186
31. Navarro-Cáceres M, Caetano M, Bernardes G, de Castro LN (2019) Chordais: an assistive system for the generation of chord progressions with an artificial immune system. *Swarm Evol Comput* 50:100543
32. Papadopoulos G, Wiggins G (1998) A genetic algorithm for the generation of jazz melodies. *Proceedings of STEP*, 98
33. Reese KW (2011) Computationally generated music using reinforcement learning
34. Roig C, Tardón LJ, Barbancho I, Barbancho AM (2014) Automatic melody composition based on a probabilistic model of music style and harmonic rules. *Knowl.-Based Syst* 71:419–434
35. Scirea M, Togelius J, Eklund P, Risi S (2016) Metacompose: a compositional evolutionary music composer. In: *International conference on computational intelligence in music, sound, art and design*. Springer, pp 202–217
36. Shan MK, Chiu SC (2010) Algorithmic compositions based on discovered musical patterns. *Multimed Tools Appl* 46(1):1
37. Ting CK, Wu CL, Liu CH (2015) A novel automatic composition system using evolutionary algorithm and phrase imitation. *IEEE Syst J* 11(3):1284–1295
38. Tokui N, Iba H et al (2000) Music composition with interactive evolutionary computation. In: *Proceedings of the third international conference on generative art*, vol 17, pp 215–226
39. Toussaint GT et al (2005) The euclidean algorithm generates traditional musical rhythms. In: *Proceedings of BRIDGES, mathematical connections in art, music and science*, pp 47–56
40. Yilmaz AE, Telatar Z (2009) Potential applications of fuzzy logic in music. In: *2009 IEEE International conference on fuzzy systems*. IEEE, pp 670–675

Publisher’s note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.