Journal of Intelligent & Fuzzy Systems 42 (2022) 4411–4423
DOI:10.3233/JIFS-219231
IOS Press

Algorithmic music generation by harmony recombination with genetic algorithm

Omar Lopez-Rincon[a], Oleg Starostenko[a,*] and Alejandro Lopez-Rincon[b]
[a]*Department of Computing, Electronics, and Mechatronics, Universidad de las Americas Puebla, Cholula, Mexico*
[b]*Utrecht University Pharmaceutical Sciences Pharmacology, Netherland*

Abstract. Algorithmic music composition has recently become an area of prestigious research in projects such as Google's Magenta, Aiva, and Sony's CSL Lab aiming to increase the composers' tools for creativity. There are advances in systems for music feature extraction and generation of harmonies with short-time and long-time patterns of music style, genre, and motif. However, there are still challenges in the creation of poly-instrumental and polyphonic music, pieces become repetitive and sometimes these systems copy the original files. The main contribution of this paper is related to the improvement of generating new non-plagiary harmonic developments constructed from the symbolic abstraction from MIDI music non-labeled data with controlled selection of rhythmic features based on evolutionary techniques. Particularly, a novel approach for generating new music compositions by replacing existing harmony descriptors in a MIDI file with new harmonic features from another MIDI file selected by a genetic algorithm. This allows combining newly created harmony with a rhythm of another composition guaranteeing the adjustment of a new music piece to a distinctive genre with regularity and consistency. The performance of the proposed approach has been assessed using artificial intelligent computational tests, which assure goodness of the extracted features and shows its quality and competitiveness.

Keywords: Automatic music composition, music feature extraction and encoding, genetic algorithm, harmony recombination

1. Introduction

In recent years, many projects around the world are emerging with companies investing in development of musical services and exploring the role of machine learning in creative processes. Among them are well-known iTunes, Spotify, Pandora, Google (AI Magenta project), Sony's CSL Lab, Aiva at Luxembourg, which perform specific oriented research in music composition [1–3]. The automatic music generation has significant economic impact and takes about 10–15% of the production budget in projects that develop feature movies, videogames, entertainment applications, commercials, TV series, etc. [3, 4]. The most common file structure used to provide music analysis is the Music Instrument Digital Interface (MIDI) supported by modern software and hardware. There are several approaches to feature extraction and MIDI analysis using semi-supervised and supervised machine learning methods, but there are not enough available structured or labeled datasets to train supervised or semi-supervised models [4–6]. There are not enough metrics or enough standard samples of the different aspects of music talking about melody, harmony, rhythm, and perceptual use of music. In addition, most methods only assist at processes of music composition and not as a whole, because several steps of music synthesis require human intervention as the creativity agent.

For generation of music compositions, a novel approach is proposed, and it can automatically extract music features from data files by abstracting

*Corresponding author. Oleg Starostenko, Department of Computing, Electronics, and Mechatronics, Universidad de las Americas Puebla, Cholula, Mexico. E-mail: oleg.starostenko@udlap.mx.

semi-structured MIDI data messages defined by music theory and reflected in the MIDI file structure. These extracted features are used as knowledge rules to generate new musical pieces applying original algorithms of harmony and rhythm recombination. In order to take advantages of artificial intelligent methods when near-solution optima is required to obtain in shorter times, the evolutionary algorithms are exploited in the proposed approach. Thus, a solution will provide more adaptable and consistent complete results invariant to length or complexity of MIDI files, as well as polyphonic music with different instruments and variable styles or genres can be created in a reasonable time about minutes or even lower. This is a scientific impact of the proposed solutions providing new forms for creating a music creator.

The rest of paper is organized as follows: in section 2 the related works are evaluated and the most common methods for music data generation are discussed. In section 3 the proposed approach is presented with a formal description and specific examples. The obtained experimental results and their evaluation are resumed in section 4. Finally, section 5 presents conclusions and future work of research in progress.

2. Related works

Despite the existence of many scientific reports, the number of approaches and methods used for development of intelligent machine music creators are quite limited. The well-known methods for new data synthesis can be subdivided into four groups such as heuristic, deep learning, stochastic and symbolic.

Among a group of heuristic composition methods, it is possible to distinguish evolutionary based algorithms and dynamic programming approaches, which successfully have been used for algorithmic music composition [7, 8]. As usual, evolutionary based approaches exploit three main concepts of genetic programming such music representation, searching guide heuristic, and evaluation of music quality. Development of evolutionary music composers is based on constrained multiple objective functions or optimal-solution search problem under constraints that reduce the subjectivity of evaluation of new music compositions [8, 9]. The melody is designed by dynamic programming and combining rhythm, chord progression, and accompaniment patterns [10]. Although several solutions based on heuristic composition methods produce quite acceptable and good audible music, they still have some disadvantages. Particularly, solutions have no guarantee to be found as well as they are sufficiently restrictive in terms of expansion of the model and operate under constrains of unique style for which they are designed. The extracted music features sometimes have simple representations using only pitch and duration accent and as consequence, the automatic composers seldom operate polyphonic music. Additionally, there are no metrics to approximate a loss function and to extract music features numerically correlated to the perception of melodies as well as generation of music is not automatic and the active presence of human in some creating processes is indispensable. Finally, the assessment of automatic music creators is provided by subjective opinions of users.

Deep learning composition methods can be subdivided into deep belief networks, convolutional networks, and recurrent networks, among which the recurrent networks are used more frequently for music generation [11, 12]. Multiple strategies for adaptation of learned features efficiently can be used in creative processes in terms of the amount of training and in terms of their ability to confront with restricted availability of learned data [13, 14]. Although these techniques provide quite successful results for spatial data analysis, generalization, and classification, there is a need in specific databases that allow not only classifying independent variables but also will consider the whole relationship between them. Therefore, the main problem with models based on neural networks is the absence of labeled datasets to learn or abstract the encoding of the music theory or the music synthesis rules.

In the group of stochastic composition methods, which are concerned with probabilistic distribution laws [15], the Markov Models and Generative Adversarial Networks (GAN) are distinguished [16, 17]. Generative adversarial nets can also be used in creative processes of music data, using the discriminative model to guide the training of the generative process and it can be considerably benefited but lacks variation. The variation, which statistical stochastic approaches provide through joint distributions, allows the generation of variables inside the correct probability density function of the task at which is trained [18]. Although deep learning and stochastic composition methods have provided significant breakthrough in development of systems for automatic music generation, there are several disadvantages, which restrict their wide application. There are not standard labeled datasets to use them with deep learning approaches directly for music generation. The reported in scientific literature systems for algorithmic creation of new data still generate monophonic music compositions, which frequently don't work in real-time as well as there are not standard quantitative metrics for evaluation how musically convincing new compositions are. Finally, human must be part of generative process at least for subjective assessing the goodness of entire work.

Several symbolic artificial intelligence methods also have been used for music generation. The agent, declarative programming, and grammar composition methods are more promising due to their high-level symbolic and human-readable representations of problems, logic, and search. Knowledge in such a model is represented symbolically and mental cognitive operations are described using symbolic structures that includes logic-based reasoning [19–21].

After analysis of relevant methods used for music generation, the convincing audible results vary significantly depending on desired outcome and level of acceptance by the user. Evolutionary algorithms are the most explored methods and there are two kinds of compositional processes: transformational or feature-based. The transformational methods allow taking already created compositions and using them as a template to make changes in the music piece with different features found in its harmony or in its rhythm. The feature-based methods select specific characteristics that could or could not be already conditional functions to create solutions out of the possible solution space. The main problem to design a feature-based method is that there is no convention in music research to establish definitive point for a specific genre or style.

The application of generative models with supervised learning methods is not recommended because they strongly depend on labeled databases, which do not exist, and the already available data is not clean, standard, or indexed. Besides the complexity and training time needed in non-linear approaches, other two problems are found. They are looping in music development or wandering, where the first refers to repetitions in automatically composed music and the second one implies melody development without any theme or rhythm. The methods based on stochastic processes have results that become repetitive without any kind of progress in music development. Lastly, the methods based on symbolic manipulation like rule-based ones need hard-coded generative knowledge-databases, which can take days for preparation [22]. Now it is possible to conclude that application of artificial intelligence approaches to music generation is very promising. Thus, the main contribution is to improve generation of new non-plagiary harmonic developments using symbolic abstraction of MIDI music with non-labeled data by applying an evolutionary based approach with controlled selection of rhythm.

3. Harmony recombination approach

3.1. Preliminary considerations

In a MIDI file represented as events, emerge three features such as *harmony, melody* and *rhythm.* While *melody* is a linear sequence of notes played at different moments, *harmony* is the use of simultaneous *pitches* (tones, notes) or *chords* played at the same time. A *melody* structure is described as *harmony* with particular *rhythm* which is a sound pattern divided into beats that repeats in time throughout music composition.

The harmony is defined by *note pitches* (a number from 0 to 11 of 12 sounds related to the frequency of a tone) and the *note octave* (difference between two notes, where one of them has twice the frequency of the other). The rhythm is defined by *note onset* (beginning of note) and *note duration* which is the length of time a pitch is sounded. So, these features can be described using only four integer values. In a MIDI file the development of musical composition is encoded by a sequence of messages or events that specify generating ordered sounds [23]. A MIDI file has binary structure that describes events measured with *ticks* as discrete representation of subdivisions of time. Each file begins by specifying the duration of the *quarter note* (*crotchet*) in ticks, which determines the speed that the music piece will have. All events described in the file are related to this value. The MIDI note number is in range from 0 to 127 however, this requires large amount of data to describe the whole music piece. If the MIDI note number is subdivided into two harmonic features such pitch and octave, the possible combinations for two notes description is reduced significantly. So, in contrast to well-known approaches [11, 24, 25], the proposed method generates new music data by mixing four main features of two different MIDI files. Particularly, a first MIDI file composition *C1* with its harmonic features *h1* is recombined with the rhythmic features *r2* of a second composition *C2* to generate a new MIDI file with common features of two source files $h1 + r2 = C3$. The

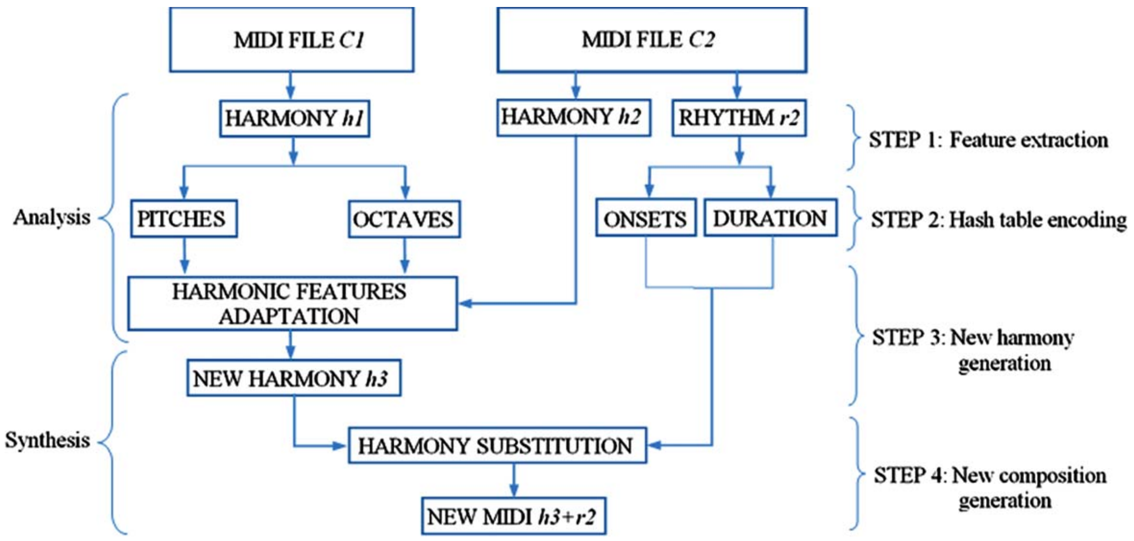

Fig. 1. Flow chart of the proposed approach for algorithmic music generation based on harmony recombination.

proposed approach is described in four steps shown in Fig. 1.

The first step which is feature extraction consists in subdivision of an entire composition into windows equal in size to the file's quarter note declared at the header of file, then each note found in every window is encoded by the corresponding integer values of pitch, octave, onset, and duration. In the second step four hash tables (dictionaries) that correspond to the four note features are generated. After the feature extraction a so-called sequence map is computed. It combines generated feature dictionaries by lines, where sequences of pitches, octaves, onsets, and durations are grouped in order of appearance for all existing windows of in the music piece. This description reduces the total number of combinations to be processed during the synthesis of the new composition. The third step replaces dictionary *h2* of *harmony* extracted from one of the MIDI files (C2) with the dictionary *h1* of harmony from the other MIDI file (C1).

This is done using a simplified genetic algorithm to take advantage of artificial intelligence in automatic music generation. Particularly, evolutionary algorithms as other AI and meta-heuristic optimization techniques are considered because music sequences don't have the same length. In the last step new created harmonic development in dictionary *h3* is combined with dictionary *r2* of rhythm of the second MIDI file resulting in a new musical composition with regularity/consistency and development. This is the principal advantage of applying a transformational compositional method, where the source composition is taken as a seed and a new composition is obtained by transforming its features [26]. The following descriptions cover the proposed approach in further detail.

3.2. *Rhythm and harmony feature extraction*

The information found in the events of MIDI file consists of a list of instruments with their numeric identifiers and a list of notes with their features (onset, duration, and pitch) to be played by each instrument. Let there be a MIDI file of a song C with length L described in ticks with a crotchet value of Q also measured in ticks. In the first step C is subdivided into (L/Q) windows of length Q. Every window W_i from C, where $0 \leq i \leq (L/Q)$, includes the sequences of notes. Each jth note in each ith window (W_i) is characterized by 4 features: pitch p_{jWi}, octave c_{jWi}, onset o_{jWi}, and duration d_{jWi}, extracted (in range from 0 to 127) by applying Equation (1).

$$p_{jWi} = N_{jWi} \bmod 12; \quad c_{jWi} = N_{jWi}/12; \quad o_{jWi} = N_{jWi}/Q \tag{1}$$

The description of composition can be subdivided into instruments using an instrument index and into sequences of windows also called words.

A word contains the information of one note with descriptions of its onset (in tics), duration (in tics), and MIDI note number shown in Fig. 2. To analyze the harmony of a composition in a MIDI file, the

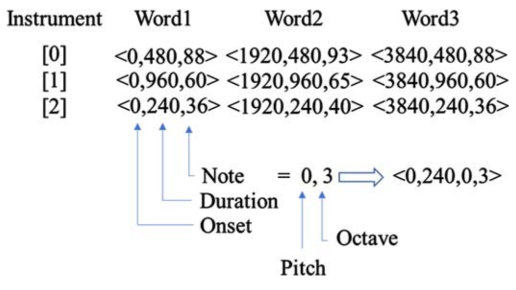

Fig. 2. Distribution of MIDI events between instruments used in composition (up) and translation of MIDI note numbers from *Word 1* to the pitches and octaves (below).

notes numbers are translated to note pitch and octave according to (1) as shown in down part of Fig. 2.

The advantages of this feature extraction process are independence from quarter note resolution in different MIDI files and it does not generate loops at selecting the best representations of MIDI note features.

3.3. *Hash table encoding*

After the subdivision of the original MIDI files into windows, a sequence of notes is found in each window, they are analyzed and used as an entry to four different hash tables (dictionaries) each one for each note feature: onsets, durations, pitches, and octaves. The second step is used to generate these note features dictionaries, which contain all the non-repeated sequences of words found in all the windows of MIDI file. The hash tables for the note pitches and note octaves, used later as description of harmony, are obtained from the MIDI events as it is depicted in Fig. 3. For example, in the first window [0] there were no sounds, in the second window [1] the word of MIDI notes < 62, 65, 69, 72 > according to pitch alphabet corresponds to notes *D*, *F*, *A*, and *C*, where first three notes are from the 5th octave and the last one is from the 6th octave. There are 11 octaves ranging from 0 to 10. Hash tables contain only non-repeated combinations of notes. So, when two similar combinations of note are appeared (i.e., windows [2] and [3] in Fig. 3), it is recorded in the dictionaries only one time.

Finally, a sequence map is generated by grouping all hash tables and consists of sequence of words found in consecutive windows with all descriptions of notes previously separated into the four dictionaries. Consider an example shown in Fig. 4, (on the left), where maps of harmonic and rhythmic features together form a sequence map.

New feature combinations of notes appear in windows from [*i*] to [*i* + 3]. These are used to extend feature dictionaries (shown in black while already described features are shown in gray). When the onsets < 0, 160, 320 > is appeared for the first time, they take next available index (1) in the onset dictionary, the previous notes have the same onset values < 0, 80, 80 > which are already recorded with index (0). The durations < 160, 160, 160 > in window [*i*] have come from a previous window, and they already are placed in the duration dictionary with index (0). Thus, the hash table of duration is updated only with new durations < 160, 160 > found in window [*i* + 1].

Finally, to simplify a sequence map and prepare it for the next step of harmony recombination by genetic algorithm, the note feature descriptions in each window are substituted by non-repeated feature word entries (indexes) from the dictionaries as it is shown in Fig. 4 (at the center in italic). For example, words for pitch, octave, onset, and duration for window [*i*] now are presented by indexes, with corresponding features stored in the dictionaries, i.e. [*i*] 3 – 2 – 0 – 2. Windows [*i*] and [*i* + 2] have the same combination

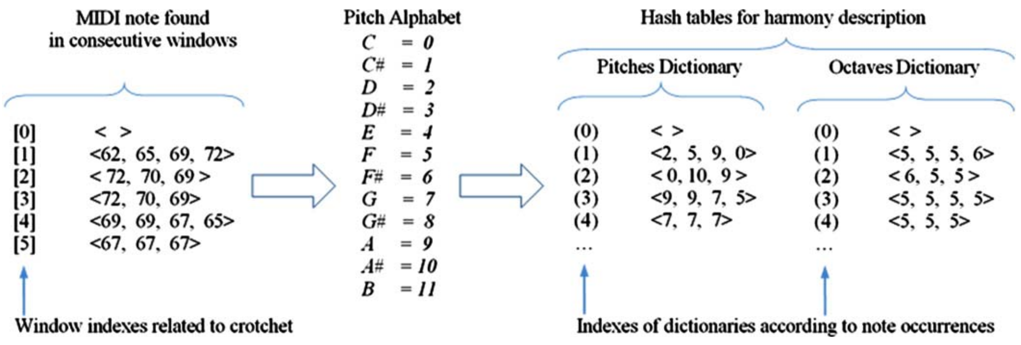

Fig. 3. MIDI notes in indexed windows (left) translated into their respective values recorded into hash tables of pitches and octaves (right).

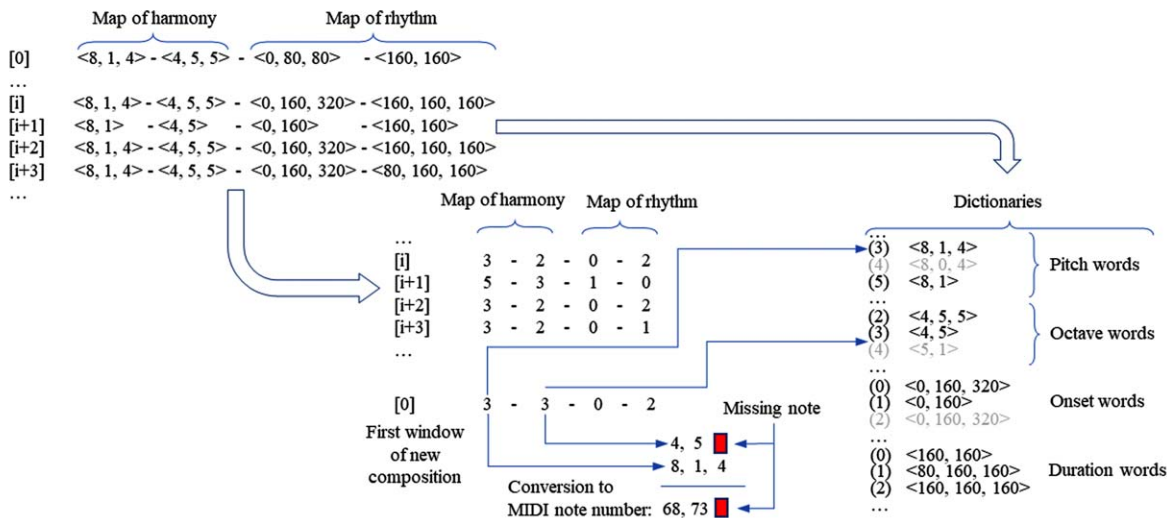

Fig. 4. MIDI events grouped into sequence map that consists of a sequence of words with note features on the left, four dictionaries with separated note features on the right and two words from dictionaries of pitches and octaves with different number of notes (below).

of notes so, in the simplified sequence map of entries they will have the same descriptions (indexes).

Applying the proposed approach, the music composition is compressed into abstract descriptors related to note features in windows specified by the MIDI file crotchet. The fact is that it does not matter how many instruments there are in a polyphonic music piece, all the windows are encoded into only four dictionaries of note features, that can be considered as another advantage of the algorithm proposed in the second step.

3.4. Generation of a new dictionary of harmony

The third step in the process of automatic music generation consists in creating new dictionary of harmony based on the pitches and octaves from a file and combining it later with rhythmic features of another music piece. Since the dictionary of harmony is a list of the non-repeating combinations of words found in a particular composition, it is different from the dictionaries of harmony of other compositions, which rhythmic features will be exploited. Inconsistencies between different dictionaries often require applying their adaptation. Therefore, to solve the discrepancies between the dictionaries the use of the genetic algorithm has been proposed also taking advantage of it in the new harmony adaptation process. This provides a measurable adjustment by fitting harmonic features to different rhythmic evolutions and assures correct audible result in new composition.

Dictionaries of different songs can vary in global features (number of words recorded in each dictionary) and in the local features (number of notes within a word). Even when using dictionaries of a single composition, words with different lengths do not fit and thus, they are not compatible with the recombination process (see Fig. 4 below). The first intention to adjust number of notes in a word is to apply their random filling or random deletion. However, fitting the length in random way can generate local inconsistencies in the harmonic development of resulting composition, because it is not possible to maintain motif or shapes, which are the periodical repetitions of the same sequence of notes. Therefore, a random decision is going to vary the result as it is shown in Fig. 5. For example, to maintain melody, the words < 68, 13, 68 > in the sequence map can be used several times. Suppose that new sequence map will be generated by recombination of pitches and octaves from their corresponding dictionaries to produce also repetitive words in new sequence map. It is possible that some pitches and octaves to be combined have different sizes and a fitting process must be applied. However, if random assignment of value X for incomplete octave word is applied, then the global consistency could be lost, because sequence length of notes in windows may be different.

For automatic music generation a new dictionary of harmonic features is proposed by recombination of feature words in the hash tables of pitches and octaves of the original music composition. Unfortunately, the idea of recombination of words in dictionary of

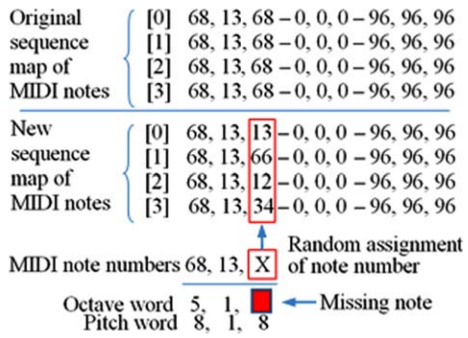

Fig. 5. Example of incompatible words of pitches and octaves and the result of random assignment of note number X causing inconsistency in melody of new composition.

rhythm is not practical, because words describing rhythmic features have different resolution in ticks and different length. However, instead of generating a new hash table of rhythm using the same original composition, the rhythmic features dictionary of another composition can be used providing adjustment of new music piece. So, for adapting refurbished dictionaries of harmonic features of a particular song to a dictionary of rhythmic features of another song, a genetic algorithm is proposed to use for forming constrained candidates of harmony, from which the best-fitted population is selected. In contrast to random adjustment of pitch and octave words, genetic algorithm is used for generating dictionaries with consistent global and local features. Using global and local metrics the adaptation problem is solved by constraining each individual of the population in size and length and then to fit into the specific dictionary of rhythm.

Let there be two compositions *C1* and *C2* each one with their four dictionaries, where pitch and octave hash tables form the harmonic section, and onset and duration form the rhythmic section. The objective is to generate a new composition *C3* by processing dictionary of harmony *h1* in order to adjust size and length of its words to *h2* and finally, to combine new generated dictionary of harmony *h3* with dictionary of rhythm *r2*. So, to generate new music piece, it is necessary to fit *h1* to *r2* assuring that the global (number of words) and local (length of words) features of new dictionary *h3* have to be the same as the global and local features of *h2*. Thus, the number of notes in every word in dictionary and the length of each word are the constraints to generate new candidates. The genetic algorithm aims to find a new set of words in the target dictionary *h3* that satisfies constraints with the minimum value of fitness function (mean square

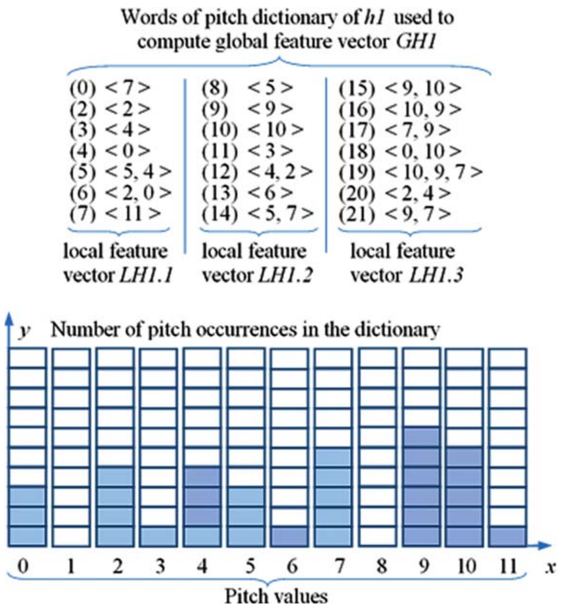

Fig. 6. Pitch dictionary divided in three columns (Top).Result of counting occurrence of pitches in pitch dictionary of *h1*, where the *x*-axis corresponds to 12 possible pitch values(Bottom).

error comparing global *GH* and local histograms *LH*). Due to a dictionary of harmony consists of pitch and octave, two separate processes for adjustment of dictionaries of pitches and octaves of *h1* to dictionaries of pitches and octaves *h2* are applied using the following algorithm.

1. The pitch dictionary of *h1* is used as a seed to equalize its number of words in pitch dictionary of *h2* by computing global histogram (*GH*) of occurrences of 12 possible pitch values. For example, if the pitch dictionary *h1* has a length of 21 words as it is shown in Fig. 6, its *GH* contains a count of 12 pitch values that correspond to pitch occurrence in the *h2*.

In the next steps of this algorithm, the occurrence of pitches in music piece will be used as probability density function (*PDF*) giving them priority to be included into newly generated music composition. The occurrence of 12 pitches ($p = 0 - 11$) can be described by the vector [3, 0, 4, 1, 4, 3, 1, 5, 0, 6, 5, 1], then the vector is normalized according to Equation (2) by dividing each pitch occurrence value *PO* by total number of pitch values found in whole dictionary *h1*.

$$PO_p = \frac{number\ of\ occurrencees\ of\ pitch\ p}{total\ number\ of\ pitch\ values} \quad (2)$$

In the example shown in Fig. 6 there are 33 pitches in the whole dictionary of 21 words. Therefore,

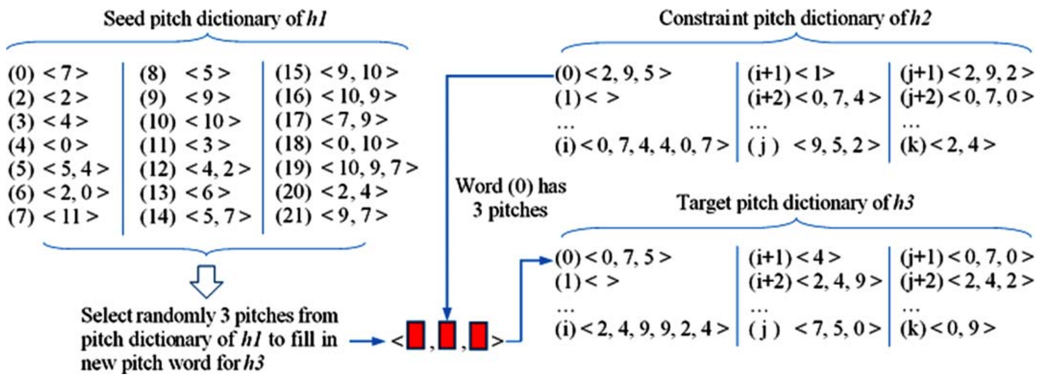

Fig. 7. Generation of words in target pitch dictionary of *h3* by substitution of pitches in dictionary of constraints of *h2* by random selection of pitches from dictionary of *h1*.

applying Equation (2), the normalized vector that represents the global histogram *GH1* of pitch dictionary *h1* is derived as *GH1* = [0.09, 0, 0.12, 0.03, 0.12, 0.09, 0.03, 0.15, 0, 0.18, 0.15, 0.03]. The vector *GH1* is the global feature (number of words in pitch dictionary) to be used by genetic algorithm adapting the size of pitch dictionary of *h1* to the size of pitch dictionary of *h2*.

2. Additionally, the pitch dictionary of *h1* is divided into three-thirds of its length, and then pitch histogram for each sub-section is also computed as it has been done in step 1 (see Fig. 6 Top). As a result, three normalized vectors of length 12, which describe three local histograms, are obtained.

Suppose that in the example shown in Fig. 6, subdivision of pitch dictionary *h1* into three sections is done as it is shown in three columns. The total numbers of pitches in the columns are 9, 9, and 15, respectively. Applying Equation (2) to each column, the following normalized vectors that represent local histograms are obtained:

LH1.1 = [0.22, 0, 0.22, 0, 0.22, 0.11, 0, 0.11, 0, 0, 0, 0.11], *LH1.2* = [0, 0, 0.11, 0.11, 0.11, 0.22, 0.11, 0.11, 0, 0.11, 0.11, 0], and *LH1.3* = [0.07, 0, 0.07, 0, 0,07, 0, 0, 0.2, 0, 0.33, 0.26, 0]. These vectors are the local features (length of words in pitch dictionary) and they are concatenated to be used by the genetic algorithm.

3. By concatenating the global and local feature vectors *GH1, LH1.1, LH1.2*, and *LH1*.3, the 48-valued vector *Y* is obtained. This vector is used for computing a fitness function using mean square value measured against another 48-valued vector obtained from the pitch dictionary of *h2*. The minimum number of sub-sections obtained from the whole dictionary of *h1* must be at least three for a start point, a middle development and ending operations in the measures of the genetic algorithm. If the bigger number of sub-sections of dictionary is used, more variations in music development can be achieved.

4. The process of feature adjustment consists in finding a solution that minimizes the difference between *k* elements of vector *Y* that represents pitch dictionary of *h1* and corresponding *k* elements of vector $\hat{Y}$ that is the generated candidate for pitch dictionary of *h3*. The generation of the candidates (also called individuals of a current population) is done by using the probability density function (PDF) of pitches in *h1* by picking them randomly from the original pitch dictionary of *h1* and to put them instead of pitches located in pitch dictionary of *h2*. It means that the pitches with higher occurrence in global histogram (see Fig. 6) have more probability to be selected for generating words in *h3*. For example, in Fig. 7 the process of generation of a new pitch dictionary *h3* is presented, where adapting words of pitch dictionary of *h1* to length of words of pitch dictionary of *h2* is done by selecting pitches from *h1* randomly but considering their occurrence in global histogram computed in step 1 of this algorithm.

After substitution of all pitches of dictionary *h2* by randomly selected pitches from dictionary *h1* a new individual of population described in $\hat{Y}$ is created. Each *k*-th element of individual $\hat{Y}$ satisfies the required global and local constraints defined by pitch dictionary of *h2*. After generating a new individual, the fitness function is a measured using mean square error (MSE) described in (3) and is computed [27]

$$MSE = \frac{1}{n}\sum_{k=1}^{n}\left(Y_k - \widehat{Y_k}\right)^2, \tag{3}$$

where Y_k is a k-th element of feature vector that represents pitch dictionary of *h1* to be compared to k-th element of feature vector $\widehat{Y_k}$ of the created individual and n is the number of elements of features in vectors Y and $\hat{Y}$.

The number of elements in feature vectors depends on the number of sub-divisions of the seed dictionary. Each sub-division will have 12 elements and 12 extra from the global histogram, for *h1* in the presented example in step 2 of this algorithm $n = 48$. The minimum value of the *MSE* for a particular individual in the current population is used as indicator that it can be used as potential candidate for generation of new musical piece.

In the proposed approach multiple populations can be generates and the best candidate will be used for generation of pitch dictionary of *h3*. However, carried out experiments show that only one population is enough to select an acceptable candidate for generation of a new music piece. Selecting the best candidate from multiple populations does not generate a music piece with audible perceived differences. Based on empirical experience, the generation of 1000 individuals in population will suffice to select a suitable candidate to fit the rhythmic dictionary *r2*. No crossing function and no mutation parameters in the genetic algorithm are required.

5. The same processes described in steps 1-4 are repeated to create an octave dictionary for *h3* with 11 possible octave values for each histogram. Finally, the dictionary of harmony *h3* is created, where pitches and octaves of *h1* are adapted to the length and size constraints of pitches and octaves of *h2*.

It is evident that different compositions will be generated every time the algorithm runs, because new dictionary of harmony *h3* is created by random variations during the adjustment of global and local features of *h1* and *h2*. The variations are reflected in the harmonic progression by changing the order of pitches, octaves, or both. Because each user expects different music development, it is recommended to run the algorithm several times to select a solution suitable for the desired outcome.

3.5. Harmonic word replacement

The last step of new MIDI file generation consists in composing the sequence map using dictionaries of harmonic and rhythmic features. The dictionary *h3* of harmonic features defined by pitch and octave dictionaries generated in the previous step is concatenated with non-changed dictionary *r2* of rhythmic features defined by onset and duration dictionaries of the composition *C2*. If Eqs. (1) have been used to extract pitch, octave, onset, and duration of each note from MIDI file message, the Equation (4) is applied to create MIDI message of new generated music piece from its sequence map that consists of combined harmonic dictionary *h3* and rhythmic dictionary *r2*.

$$N_j = c_{Wi} * 12 + p_{Wi}; o_j = Q * W + o_{Wi} \quad (4)$$

where N_j is MIDI *j-th* note identifier, c_{Wi} is number of octave and p_{Wi} is pitch value of j-th note from the i-th window W_i from the harmonic dictionary *h3*. The o_j is the onset of j-th MIDI note specified by time instant to start playing. It depends on size of window (crotchet measured in ticks), number i-th of window W in the sequence map of *C3* and onset value o_{Wi} in that window. The duration of MIDI note d_j also measured in ticks is directly specified by pitch duration d_{Wi} found in the i-th window W of the rhythmic dictionary *r2*.

It is important to mention that non-standard MIDI files cannot be directly used if rhythm of a music piece is described in channels usually assigned for harmony. Also, the final selection of the best generated music piece depends on the user evaluation of goodness and that is a subjective process, because there are no quantitative metrics for assessment of music perception.

4. Experiments and discussion

For tests of the proposed approach for automatic music generation, the *.net* framework with the *C#* language has been designed running on Windows 10 system of 64 bits using different MIDI files from the Kunst der Fuge database of classic music that consists of 19,300 MIDI files of 1000 composers [28]. The used machine has Intel(x) Xeon$^{®}$ CPU E5-2603 v3 @ 1.6 GHz with 24 GB of RAM. Particularly, three aspects are considered during evaluation. First, the precision of approach to preserve the consistent harmonic development in new compositions using mean square error is discussed as well as the access to website with collection of audible generated music compositions is provided. In addition, the complexity of the approach to generate new music pieces is computed. The quantitative metric used in genetic algorithm to approximate harmony of seed composition to harmony of generated piece is the *MSE* mean square error defined by Equation (3). The fit-

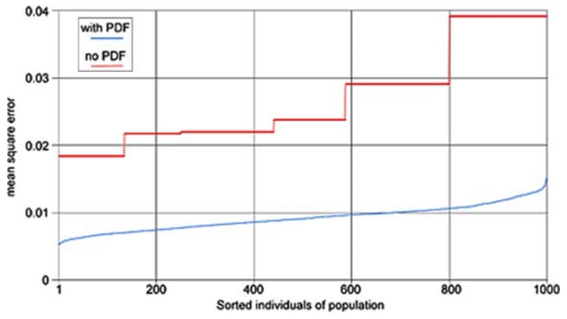

Fig. 8. MSE in uprising order computed for 1000 individuals of population, where continuous bottom line corresponds to harmonic features with PDF and upper line is obtained using only random selection of pitches excluding their occurrences in dictionary *h1*.

ness function is defined with the *MSE*, it ensures that the selected candidates used for creation of the new dictionary of harmony *h3* have the minimum difference with *h1*, generated and selected by the genetic algorithm fitness function defined in Equation (5).

$$\hat{Y}_{MMSE}(Y) = argmin_{\hat{Y}} MSE \tag{5}$$

This preserves the specific harmonic features of more frequent pitches and octaves in original composition *C1* to be used in the new music piece *C3*. Retaking the example presented in Fig. 6, the *MSE* is computed according to (3) for 1000 individuals of one population and depicted in raising order in Fig. 8. Applying the probability density function (*PDF*) in the process of selecting pitches and adapting the harmonic dictionary *h1* to *h2*, the generated dictionary *h3* is closer to the harmonic development of original composition giving minimum values of *MSE* presented by continuous blue line (below) in Fig. 8. If the generation of words in target harmonic dictionary *h3* is made by random selection of pitches and octaves from dictionary of *h1* without considering the probability of their occurrences found in the global histogram *GH1*, the *MSE* values indicate the worst fitting of harmonic features of original composition to new one (see upper red line in Fig. 8).

It can be observed that using only one population with no more than 1000 individuals the fitness value between harmonic dictionaries of *h1* and *h3* achieves *MSE* of 0.005. The low values of the *MSE* correspond to the case, when pitches adapted to new dictionary *h3* are selected from any word of entire harmonic dictionary *h1* i.e. using vector of global and local features.

In the following site https://open.spotify.com/artist/4zATrFzcT0PBh8vSl49gAJ the reader can find a collection of generated music pieces created by applying the proposed approach to MIDI classic music from Kunst der Fuge database [28]. The user can appreciate high quality polyphonic music played on poly virtual instruments, where each new original composition has been created by the recombination of harmonic features of different music pieces preserving the rhythmic features of the composition used as development constraints.

The proposed approach has low computational complexity, it does not require significant amount of data for abstraction and no training is needed for music feature encoding. In contrast, the well-known methods for music generation are based on neural networks with more than two layers to cluster the information used to learn or abstract the encoding of the music theory [29]. The time of generation of the harmonic and rhythmic dictionaries can be described by a constant u specified by the number of crotchet windows in a music piece and the number of notes in each window. Computing global and local feature histograms can be measured by the constant h. Then a feature vector of size s computed from global and local histograms is compared with generated n vectors of $\widehat{Y_x}$ corresponding to n individuals of population U using *MSE* in Equation (3). The next step is the adjustment of *MSE* for n individuals in raising order by using quicksort algorithm, which in the worst-case scenario gives $\mathcal{O}(n^2)$. Finally, the time for creation a MIDI message from generated dictionaries of harmony *h3* and rhythm *r2* using Equation (4) can be specified by the constant v. Therefore, the whole complexity of the proposed approach for automatic generation of music composition is equal to $\mathcal{O}\left(n^2 + (U * (n * (p + h + s)))\right.$ $\left.+v\right)$.

The last set of tests has been carried out to analyze how relevant the music features are in the proposed approach. For this test 15 compositions of classical composers particularly, of Beethoven and Bach, have been selected, and their global and local histograms were extracted applying our method. Therefore, the dataset of features of these 30 compositions has been created adding labels of the composer to which the music piece belongs. The implementation of test has been run on *scikit-learn*, which is a standard framework used for machine learning in Python [30], where the ten-fold cross-validation is applied as a standard machine learning metric for evaluating classifiers. The idea is to feed the multiple classifiers by features from the dataset to detect if they can learn and correctly rank the input data.

Table 1
Ten-fold cross-validation applied to different classifiers

	Mean	Variance
Gradient Boosting Classifier	0.7333	0.3266
Random Forest Classifier	0.8000	0.2211
Logistic Regression	0.8000	0.2211
Passive Aggressive Classifier	**0.9000**	**0.1728**
SGD Classifier	0.8667	0.2211
SVC (linear)	0.8333	0.1667
Ridge Classifier	0.7333	0.2494
Bagging Classifier	0.8333	0.1667

Table 2
Confusion matrix from ten-fold cross-validation with passive aggressive classifier

	Beethoven	Bach
Beethoven	14.0	1.0
Bach	2.0	13.0

The dataset is divided into 10 groups, where 9 are used to train the classifier and the 10-th is used for classification test. This process has been performed 10 times and the results of computing mean and variance of classification of input data for each classifier are presented in Table 1. The obtained results show that all classifiers pass the validation test assuring that the used dataset contains relevant information. Taking the results of 10 tests performed by the best *Passive Aggressive Classifier*, the confusion matrix of recognizing two composers is obtained as shown in Table 2. It indicates that out of 15 pieces of Beethoven 14 have been classified correctly, while 13 pieces of Bach also have been recognized as well. This test assures that feature vectors obtained from the global and local histograms of the proposed approach have relevant information to be recognized even though not so robust classifiers were used.

After analyzing the results obtained by applying the proposed approach, some outcomes should be considered for designing similar frameworks.

– The quality of the generated music pieces depends on the quality of original extracted data used for abstraction and recombination.
– Special events in MIDI file can interrupt or generate errors of continuity during music files recombination. To avoid errors, the step of cleaning data should be applied to improve data quality and to speed up music generation process.
– There is practical use of non-linear methods like neural networks with a limited size of generated files. The learning space must be reduced to decoding octaves and pitches and so, less training would be needed for music composer.
– The duration of the used music pieces can affect the results in tone variations by making the new compositions harmonically repetitive. It needs to measure how suitable are two files to be recombined or to specify extensions for variation of the shortest file.
– When a file emulates several different instruments on one event channel, it will be easy to control and maintain the complexity of the feature structures, because a generated sequence map for each instrument shares the same dictionaries. This reduces time for recombination of dictionaries and assures that the polyphony keeps working in new generated compositions.
– The created music pieces can be complete music compositions and templates for generating music variations in multiple MIDI files.

5. Conclusions and future work

The main result of this research is the development of an original approach used for music feature extraction from symbolic representation of sounds in MIDI files with no labels and specific standards, as well as the process of generating new music compositions based on the recombination of harmony and rhythm of multiple files. Currently, the MIDI files in paid databases and free repositories from different websites are very popular but at the same time, it is a challenge to find common features inside the symbolic music descriptions, because there are no standard definitions of file structures required for their normalization, processing, analysis, and modification.

In the proposed approach the music features are abstracted and encoded to use them with a quarter note as a self-descriptor which is inside the MIDI file declared only one time in every file. This allows quantizing the music events and grouping them into the abstractions used for generation of feature dictionaries related to unique crotchet value of a MIDI file. It was possible to extract harmony and rhythm of music file and then subdivide them into pitch, octave, onset, and duration. Therefore, the theoretical contribution of this paper is a novel technique for music feature encoding and synthesis of new pieces by recombination of MIDI files entirely autonomously with no human interaction during the compositional process or at the validation steps. Particularly, the generation process consists in the adjustment of harmony of a seed music piece to size and length of harmonic features of another composition, which rhythmic development is used for adapting pitches and octaves and their recombination. In order to take advantage of artificial intelligence in automatic music generation, the genetic algorithm has been applied for the process of adapting harmonic and rhythmic features of multiple music compositions providing the creation of complete polyphonic and poly instrumental composition without user intervention. Therefore, the interaction with user is required only during the selection of a particular desired outcome form multiple music compositions generated after running the approach several times. This constrained was imposed as part of the specific goals of the research so that the resulting methods would not be biased by the user selections or his preferences including the rendering results.

Due to there are not quantitative standard metrics to measure the quality of generated music, the evaluation of the proposed approach has been done using computational testing, which validate the quality of the feature extraction and encoding processes and assure the goodness of entire generated music piece. In order to let the reader have his proper subjective opinion about the quality of the created music compositions made by the proposed approach, the generated music pieces have been recorded into collection found in https://hitrecord.org/users/procopio as well as they also are placed to *Spotify* https://open.spotify.com/artist/4zATrFzcT0PBh8vSl49gAJ.

In the future work, the development of new generative models could benefit from the creation of profiles of songs based on the proposed approach for feature extraction and music piece creation. A weighted version of the time windows at each moment could be developed taking into consideration the complete sequence of the tones using as metric the cosine distance to generate the self-similarity matrixes. This abstraction can help to precise better the composition profile, which could be used to design an objective function by finding the global optima exploiting meta-heuristics like genetic algorithm or simulated annealing. Both the profile and meta-heuristics can serve to implement a more efficient generative model. In addition, based on the presented approach, it is possible to create a composer recognition technique based on dimensionality reduction of harmonics.

We already have some promising results for the solution of the mentioned problem. The subdivision of the symbolic composition could be translated to music signal processing by using wavelets or window short time Fourier transform as well as the analysis of self-declared beat in the composition in frequency domain and it can be used to find thresholds of differences from excerpts of audio. Another important research derived from the proposal is the analysis of rhythmic statistical behavior. It is possible to derive the results at different dictionary resolutions and obtain a rhythmic development, preserving polyphony instead of constrained to composition duration other than the imposed by the user.

The results of this research can help to determine new architectures used for analysis of data relevant features as the atomic values of information to abstract them. By achieving these objectives, we are a step forward into the automatic generation of data and settle the basis for new creative agents to help society move beyond actual expectations of imagination. It will evolve into new possible businesses and next generation of entertainment satisfying curiosity by allowing a broader community to experiment, create, and invent.

Acknowledgments

This work was made possible thanks to the partial funding of the UDLAP and CONACYT, Mexico.

References

[1] A.I. Google, Magenta – open source research project. (June 2021), https://ai.google/research/teams/brain/magenta/

[2] AIVA, The Artificial Intelligence composing emotional soundtrack music. (July 2021), https://www.aiva.ai/

[3] ThinkSync Music, A music supervision company: Music in films: budgeting continued. (July 2021), http://thinksyncmusic.com/guides/music-in-films-budgeting-continued/

[4] O. Lopez-Rincon and O. Starostenko, A simplified generative model based on gradient descent and mean square error, in: Technology, Science and Culture – A Global Vision, S. Picazo-Vela, L. R. Hernández, eds., Intech, London, UK, 2019, pp. 119–127, DOI: 10.5772/intechopen.90099.

[5] M. Grachten and C.E.C. Chacón, Strategies for conceptual change in convolutional neural networks, CoRR, abs/1711.01634 (2017), http://arxiv.org/abs/1711.01634.

[6] M.A. Kaliakatsos–Papakostas, et al., Interactive evolution of 8-bit melodies, in: Proc. Int. Conf. on Evolution and Biolog. Inspired Music and Art, Malaga, Spain, 2012, pp.141–152.

[7] A. Eigenfeldt, Corpus-based recombinant composition using a genetic algorithm, *Soft Computing* **16**(12) (2012), 2049–2056.

[8] A. Eigenfeldt, O. Bown, A.R. Brown and T. Gifford, Flexible generation of musical form: beyond mere generation,

in: Proc. 7th Conf. on Computer Creativity, Paris, 2016, pp. 264–271.

[9] M. Scirea, J. Togelius, P. Eklund and S. Risi, MetaCompose: a compositional evolutionary music composer, in: Proc. Conf. on Evolution and Biol. Inspired Music and Art, Porto, Portugal, 2016, pp. 202–217, DOI:10.1007/978-3-319-31008-4_14.

[10] S. Fukayama, K. Nakatsuma, S. Sako, T. Nishimoto, and S. Sagayama, Automatic song composition from the lyrics exploiting prosody of Japanese language, in: Proc. 7th Sound and Music Comp. Conf., Barcelona, Spain, 2010, pp. 1–4, https://zenodo.org/record/849727#.XlVXQ6hKhPY.

[11] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. MIT Press, 2016. https://www.deeplearningbook.org/.

[12] D.D. Johnson, Generating Polyphonic Music Using Tied Parallel Networks, in: LNCS 10198, Comp. Intel. in Music, Sound, Art and Design, J. Correia, V. Ciesielski, and A. Liapis, eds. Cham: Springer International Publishing, 2017, pp. 128–143.

[13] G. Bickerman, S. Bosley, P. Swire, and R.M. Keller, Learning to create jazz melodies using deep belief nets, in: Proc. 1st. Conf. on Comp. Creativity, Lisbon, Portugal, 2010, pp. 228–237.

[14] A. van den Oord, N. Kalchbrenner and K. Kavukcuoglu, Pixel recurrent neural networks, arXiv:1601.06759 [cs], (2016), http://arxiv.org/abs/1601.06759.

[15] H. Järveläinen, Algorithmic musical composition, in Seminar on content creation Art@ Science, Helsinki: University of Tech., Lab. of Acoustics and Audio Signal Proc., 2000, pp. 1–11.

[16] M. Ackerman and D. Loker, Algorithmic songwriting with ALYSIA, CoRR (2016), http://arxiv.org/abs/1612.01058

[17] A. Papadopoulos, P. Roy and F. Pachet, assisted lead sheet composition using FlowComposer, in: Proc. Conf. on Princ. and Pract. of Constraint Program., Toulouse, France **9892** (2016), 769–785. DOI: 10.1007/978-3-319-44953-1_48.

[18] L. Yu, W. Zhang, J. Wang and Y. Yu, SeqGAN: sequence generative adversarial nets with policy gradient, in: Proc. 31st Conf. on Artif. Intel., San Francisco, California, USA **4** (2017), 2852–2858, http://arxiv.org/abs/1609.05473

[19] M. Navarro, J. Corchado and Y. Demazeau, A musical composition application based on a multiagent system to assist novel composers, in Proc. 5th Conf. on Comp. Creativity, Slovenia, 2014, pp. 1–5.

[20] D. Quick and P. Hudak, A temporal generative graph grammar for harmonic and metrical structure, in: Proc. Comp. Music Conf., Perth, Australia, 2013, pp. 177–185.

[21] D. Quick, Learning production probabilities for musical grammars, *Journal of New Music Research* **45**(4) (2016), 295–313.

[22] O. Lopez-Rincon, O. Starostenko and G.A.-S. Martin, Algorithmic music composition based on artificial intelligence: A survey, in Proc. Conf. CONIELECOM., Puebla, Mexico, 2018, pp. 187–193, DOI: 10.1109/CONIELECOMP.2018.8327197.

[23] D. Back, Standard MIDI-File Format Spec. 1.1, (July 2021), http://www.music.mcgill.ca/ ich/classes/mumt306/StandardMIDIfileformat.html.

[24] M. Modrzejewski, M. Dorobek and P. Rokita, Application of deep neural networks to music composition based on MIDI datasets and graphical representation, in: LNCS, 11508, Artificial Intelligence and Soft Computing, L. Rutkowski, et al., eds. Cham: Springer International Publishing, 2019, pp. 143–152.

[25] N. Hewahi, S. AlSaigal and S. AlJanahi, Generation of music pieces using machine learning: long short-term memory neural networks approach, *Arab Journal of Basic and Applied Sciences* **26**(1) (2019), 397–413.

[26] L. Bigo and D. Conklin, A viewpoint approach to symbolic music transformation, in LNCS 9617, Music, Mind, and Embodiment, R. Kronland-Martinet, M. Aramaki, ed. Cham: Springer, 2016, pp. 213–227. DOI: 10.1007/978-3-319-46282-0_13.

[27] X.R. Li and Z. Zhao, Measures of performance for evaluation of estimators and filters, in: Proc. SPIE Conf. on Signal and Data Processing of Small Targets, San Diego, USA, 2001, pp. 1–12, DOI: 10.1117/12.492751.

[28] Kunst der Fuge, The largest resource of classical music in.mid files (2020), http://www.kunstderfuge.com/.

[29] D. Meredith, Compression-based geometric pattern discovery in music, in Proc. 4th Workshop on Cognitive Information Processing, Copenhagen, Denmark, 2014, pp. 1–6, DOI: 10.1109/CIP.2014.6844503.

[30] G. Varoquaux, et al., Scikit-learn: machine learning without learning the machinery, *GetMobile: Mobile Comp. and Com.* **19**(1) (2015), 29–33. DOI:10.1145/2786984.2786995.