# Evaluation Rules for Evolutionary Generation of Drum Patterns in Jazz Solos

Fabian Ostermann(✉), Igor Vatolkin, and Günter Rudolph

Fakultät für Informatik, Technische Universität Dortmund, Dortmund, Germany
{fabian.ostermann,igor.vatolkin,guenter.rudolph}@tu-dortmund.de

**Abstract.** The learning of improvisation in jazz and other music styles requires years of practice. For music scholars which do not play in a band, technical solutions for automatic generation of accompaniment on home computers are very helpful. They may support the learning process and significantly improve the experience to play with other musicians. However, many up-to-date approaches can not interact with a solo player, generating static or random patterns without a direct musical dialogue between a soloist and accompanying instruments. In this paper, we present a novel system for the generation of drum patterns based on an evolutionary algorithm. As the main extension to existing solutions, we propose a set of musically meaningful jazz-related rules for the real-time validation and adjustment of generated drum patterns. In the evaluation study, musicians agreed that the system can be successfully used for learning of jazz improvisation and that the wide range of parameters helps to adapt the response of the virtual drummer to the needs of individual scholars (Examples of generated music are available at http://sig-ma.de/wp-content/uploads/2017/01/JazzDrumPatterns.zip.).

**Keywords:** Evolutionary music generation · Rhythm generation · Jazz solo accompaniment

## 1 Introduction

To play like Miles Davis or John Coltrane is the dream of many amateur and professional jazz students around the world. But that requires many years of experience and practice. The task is so complex, because for jazz improvisation the soloist has to spontaneously invent melodies in his/her mind suitable to background music and play them immediately, matching the solo to the tones played by other members of a band at the same time: from 3–5 musicians in a combo up to 10 and more in a big band. In the end, the successful practice of improvisation comes down to learning by doing.

For learning of improvisation without a band, the accompanying musicians must be replaced by some technical solution. A common approach is to use a prerecorded music track called *playalong* to which the scholar can play. A widely spread collection of playalongs is the *Jamey Aebersold Play-A-Long Series* [1]


J. Correia et al. (Eds.): EvoMUSART 2017, LNCS 10198, pp. 246–261, 2017.
DOI: 10.1007/978-3-319-55750-2_17

that exists since 1967. A currently popular collection can be found online at *learnjazzstandards.com* [2].

In the 1990s, a more versatile way of background music generation was presented by PG Music when releasing the software *Band-in-a-Box* [3]. Its algorithms generate music according to adjustable stylistic rules. As an input, they get a chord progression, time signature, and tempo, therefore enabling the creation of customized playalong tracks with home computers. A currently popular mobile application with similar features is *iReal Pro* [4].

The limits of such systems is that the generated tracks only differ from each other because of rules based on random variables. They can not react to improvised melodies of the soloing musician, which is necessary to learn the interaction in a professional jazz band.

In this paper, we present a novel evolutionary-based approach for the generation of accompaniment with a real-time adaptation to the currently played jazz solo. The evaluation and adaptation are based on the set of rules with regard to desired properties of jazz accompaniment. This provides an opportunity to simulate a band during the practicing of improvisation in order to gain learning effort. The initial implementation is restricted to the generation of drum patterns only, but can be extended to other instruments in the future taking into account melodic and harmonic rules.

In the next section, we first provide a brief overview of related works. Section 3 presents the representation and mutation operators of the proposed evolutionary system for music generation, and its implementation is described in Sect. 4. The rules for an automatic evaluation of the system are listed in Sect. 5. The results of the application and evaluation of the system are discussed in Sect. 6. The conclusions and ideas for future work are provided in Sect. 7.

## 2 Related Works

Below, we list some works on jazz and drum generation ordered by the impact they had on developing the system presented in Sects. 3 and 4.

First, the *Genetic Jammer* (GenJam) [5] proposed in 1994 by Biles shall be mentioned. It models a virtual jazz student who learns how to improvise by getting feedback from a human mentor whether played melodies are good or bad. This feedback is used to rate melody individuals in an evolutionary algorithm. The special use of jazz harmony theory and fixed rhythmic patterns to create rhythm-based jazz music was inspiring for our system.

Another very creative work on jazz improvising and interactive systems is *Voyager* [6], introduced by jazz trombonist George Lewis in 1986. Lewis calls it a *virtual improvising orchestra* which interactively generates musical responses to analyzed jazz performances as well as own creative musical impulses in a musical dialog with a human jazz soloist. The concept of its performance analysis was fundamentally inspiring for our approach.

Yee-King tries to explore the variety of timbres of percussion instruments with an *evolving drum machine* [7]. He uses evolutionary computing to alter the sound of drums. The concept of the drum machine is adopted in Sect. 3.

The *NEAT drummer* [8] by Hoover and Stanley uses a neural network called *NeuroEvolution of Augmenting Topologies* (NEAT) for evaluating individuals that represent drum patterns. Their focus on balancing structure and innovation of composed drum accompaniment is elementary for creative systems.

The system *CONGA* (Composition in Genetic Approach) [9] by Tokui and Iba tries to generate rhythmic compositions of rhythmical patterns evaluated by humans. It is shown that genre-specific rhythms can be produced on purpose.

*Sbeat* [10] by Unemi and Nakada does something very similar with compositions consisting of guitar, bass, and drums. The music individuals are interactively evaluated by human. The system is intended to assist beginners in composing music they favorite.

The *GeneticDrummer* [11] by Dostál generates a human-like drum accompaniment for a given score of an instrument such as guitar, bass, or piano by means of evolutionary computing. A drum stroke is not only defined by its time, velocity, and percussion instrument, but also by the playing style, e.g., where to hit the drums and how to hold the stick.

## 3 Evolutionary System: Representation and Mutation

In our proposal, each solution represents a single musical bar of a *drum pattern* which is a combination of single patterns to be played on individual instruments of the drum set. Drum patterns contain a fixed number of tics and are stored in a two-dimensional array as shown in Fig. 1. For example, a 4/4-bar with eighth notes can be represented by eight tics. The six instruments with individual patterns are bass drum, hihat, snare, ride cymbal, a low tom, and a high tom. A cell of the drum pattern array contains an integer value in range 0–127. This range is adopted from the MIDI[1] specifications. If the value is greater than 0, then it is called an *active cell*. An active cell represents a drum stroke, i.e. a hit with a (wooden) drum stick, with a given strength represented by its loudness value. The higher the value is, the louder the played instrument sounds. A recognizable drum pattern is formed by all its active cells and the corresponding instrument sounds.

Because all patterns will descend from the same user defined initial pattern, we apply no recombination. Only a simple mutation operator is used for evolution. It randomly selects a cell of a drum pattern and sets its value to a random integer number in range 0–127. That is an analogy to how a manual manipulation of a classic drum machine would proceed.

The probability that an active cell will be switched off after the mutation is equal to $\frac{1}{127} < 8‰$. Because the evolutionary process must be able to decrease the density of its patterns for musically reasons, the probability for rests must be increased. Assuming that very silent strokes are not mentioned by a listener or are even disturbing, all cells with a loudness value below 32 are set to 0. This

[1] *MIDI* (Musical Instrument Digital Interface) is a technical standard that allows electronic musical instruments, computers, and other related devices to share musical information with one another. For detailed information, see [12].

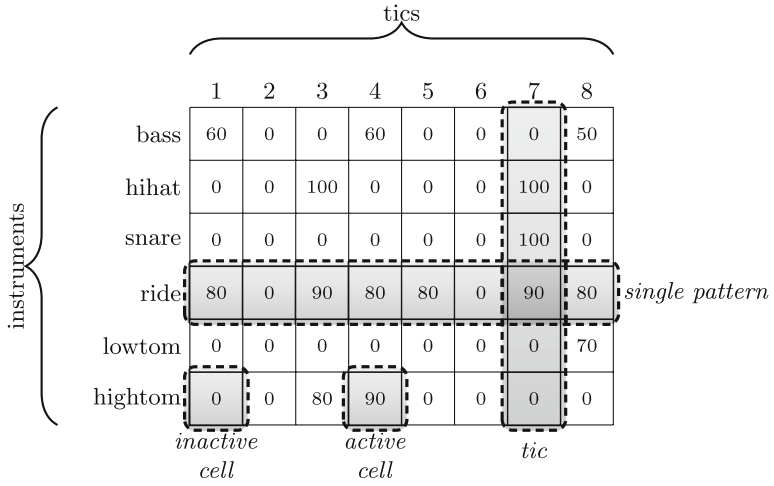


**Fig. 1.** Drum pattern represented by an array

changes the probability of raising the number of inactive cells in a drum pattern to $\frac{32}{127} \approx 25\%$.

The extent of variation in the patterns should be under control of the user, as sometimes quick changes of the accompaniment are musically desirable. Therefore the variation strength is made adjustable by the number of times the mutation operator is applied to a drum pattern. The parametrization of this mutation strength is carried out indirectly by choosing the number of generated offspring $\mu = 2^k - 1$ with $k \in \mathbb{N}$. Since small changes should be more frequent than large changes, $2^i$ individuals of the offspring population are mutated $k - i$ times for $i = 0, \ldots, k$.

## 4 Evolutionary System: Implementation

The environment of the system is sketched in Fig. 2. Human interaction with the system is marked with a 웃-sign. The $웃_{musician}$ playing the solo is typically identical with the $웃_{user}$ who controls parameters of the system.

In an application scenario, a musician is soloing on an instrument that is able to produce MIDI events. The stream of these events is then evaluated by the system, which also produces an outgoing MIDI event stream for a drum synthesizer. The sound of the synthesizer is interpreted by the musician as accompaniment to the solo. In case there is any audience, the sounds of the musician and the synthesizer are interpreted as a compounded musical performance.

A musically meaningful MIDI drum event stream must be produced in real-time by analyzing a MIDI stream from a human musician's jazz solo. To fulfill that task, the system is split up into three modules named *input module*, *genetic module*, and *playback module*, as visualized in Fig. 3.

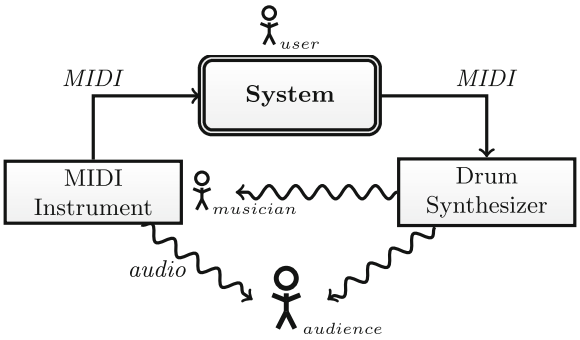


**Fig. 2.** The evolutionary system's environment

The input module manages the incoming MIDI stream and stores it in a buffer called *input window*. Details are provided in Sect. 4.1. The input window is the basis for the musical analysis carried out by the genetic module. This module generates drum patterns by means of an evolutionary algorithm as described in Sect. 4.2. The generated drum patterns are in turn taken by the playback module to create the output MIDI stream as discussed in Sect. 4.3.

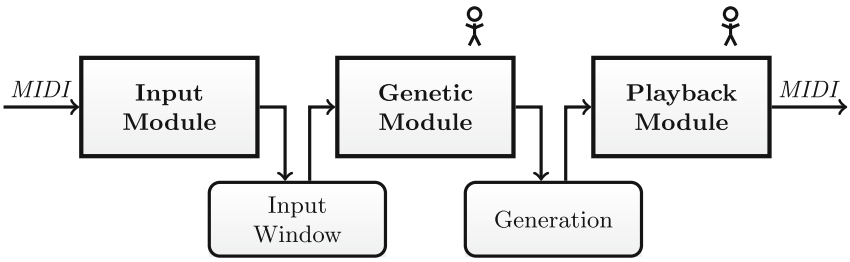


**Fig. 3.** Modularisation of the evolutionary system

### 4.1 Input Module

The difference between MIDI events and a musical note representation is that MIDI events are created to describe the actions of a piano player. So, the important actions are key pressing (*NoteOn* event) and releasing (*NoteOff* event). The input module translates these events into a representation of musical notes that will be easier to analyse. After a *NoteOn* event of a certain key is received, it is waited for the *NoteOff* event of that key. Then, a note record is written to the input window with attributes key number, velocity, start time, and length. These attributes are the basis for the analysis carried out by the genetic module.

The problem of waiting for a *NoteOn*/*NoteOff* pair is the delay when taking the record. Single *NoteOn* events are ignored in the solo analysis. It will be assumed, that in a monophonic jazz phrase the notes are short enough to not delay the analysis gravely. With this practical limitation no presumption of note lengths must be processed.

When using an Audio-MIDI-Converter, nearly every melodic instrument can be used as input instrument. Corrupted MIDI events coming from misinterpretation of the digitized audio signal can be ignored because of the goal to generate creative drum accompaniment. In the following, we will see that if the MIDI input is slightly noised by the use of a converter, this will not distort the musically meaning of the produced drum response.

The way of the MIDI event through the input module is shown in Fig. 4.

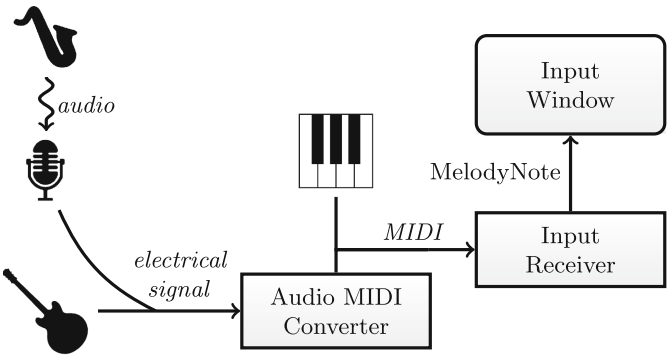


**Fig. 4.** Schematic illustration of the input module

### 4.2 Genetic Module

The genetic module is the main component of the system which operates with the help of an evolutionary algorithm. The loop starts with the initialization of the first population. It is filled with copies of user-defined initial drum pattern explained in Sect. 3. The population size is also defined by the user and can be adjusted during a session. Afterwards, the mutation is done as discussed in Sect. 3. Fitness values are assigned to individuals with the help of predefined rules based on musical theory which are explained in detail in Sect. 5. These rules can be weighted with regard to user preferences. Finally, individuals with highest fitness values are selected from the offspring population until the given population size is reached.

The evolutionary loop continues until the user stops the session. After the selection, the loop delays for a user-defined *sleep time* thus influencing the response time of produced accompanying drum patterns. If the sleep time is too short for the calculation of fitness values, the user is warned but the calculation is not interrupted.

### 4.3 Playback Module

The playback of the current population is separated from its generation by the genetic module. There is no adaptation of rhythmical pulse from the human improvised solo. The created musical timing is based on the principle of a classic drum machine. A metronome sends an increasing tic command to the pattern player which fetches a pattern at random from the current generation and initiates to play all active cells of the tic by sending MIDI events to a drum synthesizer. Figure 5 illustrates this procedure.

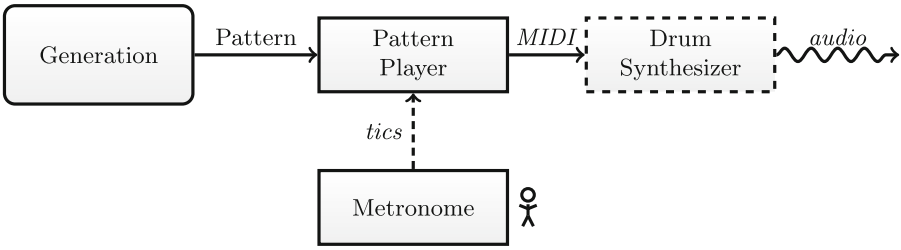


**Fig. 5.** Schematic illustration of the playback module

If the metronome tic increases to be greater then the defined length of the patterns, the following tic command is again set to 0. At the latest to that moment, a new pattern is fetched from the generation. Because the evolutionary loop can be faster than the playback of an individual pattern, the pattern player checks for a new generation on every tic. If there is a new generation created, a new pattern is immediately taken to be up-to-date with the evolutionary process.

The tempo of playback can be manually controlled by setting the tics-per-minute value (TPM) which defines the time to rest between tic commands as

$$t_{\Delta} = \frac{60\ 000\ \text{ms}}{TPM}. \tag{1}$$

Note that the patterns grid is not limited to a specific note values level. It can be used for eighth note, 16th note or even triplet patterns by appropriately increasing the TPM value and the pattern grids length.

Additionally, the metronome offers to play the tics as *shuffle* rhythm. Shuffle means a time shift of every second note by a constant factor. It is a typical stylistic feature of swing and funk and is essential for jazz music feel. Mathematically, every second waiting time $t_{\Delta,2}$ is shorter than the prior one $t_{\Delta,1}$. The shuffle factor $f_{shuffle}$ implies the formulas for the waiting times as

$$t_{\Delta,1} = \frac{2 \cdot f_{shuffle} \cdot 60\ 000\ \text{ms}}{TPM} \tag{2}$$

$$\text{and } t_{\Delta,2} = \frac{2 \cdot (1 - f_{shuffle}) \cdot 60\ 000\ \text{ms}}{TPM} \tag{3}$$

$$\text{with } f_{shuffle} \in [0.5, 1] \Rightarrow t_{\Delta,1} \geq t_{\Delta,2}. \tag{4}$$

By adjusting those settings, the user is able to manipulate the playback as desired to support his or her creative process.

## 5 Weighted Evaluation Rules

The evaluation of individual patterns is based on heuristic features calculated from an excerpt of the input window. The excerpt ignores notes being played too far in the past. The time difference is called *input window size* and can be manually adjusted. The input window size influences the temporal memory length of the virtual drummer. The longer it is, the more musical context can be taken into account. That obviously increases the calculation time. A short memory lets the drummer respond faster, but a too short one cannot bear any meaningful information.

The following features are determined from the input window excerpt:

1. $\#_{\mathbf{notes}}$ – total number of notes
2. $\#_{\mathbf{pitches}}$ – number of different pitches
3. $\#_{\mathbf{pitches_{12}}}$ – number of different pitch classes (max. 12)

Additionally, minimum $MIN(x)$, maximum $MAX(x)$, range $R(x)$, arithmetic average $\overline{x}$, and standard deviation $\sigma(x)$ are calculated over all notes for following note attributes:

1. **length (leng)**: Duration of a note
2. **keynumber (key)**: Mapping of a note to a key of the piano
3. **volume (vol)**: Loudness of a note
4. **interval (int)**: Interval between two consecutive notes
5. **distance (dist)**: Duration between the start time of two consecutive notes
6. **gap**: Duration between end time of a note and start time of the consecutive note

Evaluation rules rate drum patterns with the help of these statistic features concerning their musical compatibility to solo excerpts. Each rule describes an independent musical task the virtual drummer should fulfill. Each drum pattern in the current population is once rated by each rule, and the overall fitness is estimated as the sum of those single ratings.

Before the rules are estimated, some conditions are typically checked. If a condition is negative, the rating of the rule is set to 0. Otherwise, a solo factor $f_s$ and a pattern factor $f_p$ are calculated to describe the intensity of a rule characteristic in a solo excerpt and a drum pattern. $f_s$ and $f_p$ take values between 0 and 1, where 0 corresponds to a low intensity and 1 to a high one. $f_s$ is based on the statistic features only. To calculate $f_p$, the drum pattern array is analysed. The overall fitness value is a combination of both factors.

The design of the rules is the most subjective property of the whole system, as they define fundamentally the behavior of the virtual drummer. Therefore, the user has the possibility to define weights of rules $w \in [0, 1]$ which are multiplied with the ratings of rule functions. With this procedure, some rules may gain more influence or be completely deactivated.

The three groups of rules (*keep*, *reaction*, and *random*) are explained in the following subsections.

### 5.1 Keep Rules

The rating of keep rules is based on the distance between a current drum pattern and a user-defined initial pattern. They ensure that the patterns of the population keep a musical link to the initial one. In contrast to reaction rules described in Sect. 5.2, no conditions are checked before the rule estimation. If the solo is pausing and consequently no reaction rule is estimated, the patterns return to the initial pattern. This produces a musical relaxation allowing for the necessary contrast between a low and a high intensity in the musical dialogue between a soloist and a drummer. The keep rules do not calculate a solo factor which means they rate regardless of the solo. Therefore, the fitness is just the value of the pattern factor.

In the following, $P_{i,j}$ is the array of the drum pattern to rate, where $i \in \{0, \ldots, I-1\}$ with the number of tics $I$, and $j \in \{0, \ldots, J-1\}$ with the number of instruments $J$. Further, $A_{i,j}$ is the array of the initial pattern.

**KeepOriginal Rule.** This rule measures the distance between a pattern $P$ and the initial pattern $A$. The greater the distance is, the lower is the fitness. For every cell, the distance is calculated by the absolute difference between the cell values. For normalization, the difference is divided by the largest possible distance.

$$\textit{Pattern factor} \qquad f_p = 1 - \sum_{i=0}^{I-1} \sum_{j=0}^{J-1} \frac{|P_{i,j} - A_{i,j}|}{MAX\{A_{i,j}, 127 - A_{i,j}\}}$$

**KeepInstruments Rule.** Patterns, which use the same instruments as the initial pattern, are rated higher than others. The idea is that similar instrumentation will lead to similar sound impressions, so that a musical connection between such patterns is recognizable.

$$\textit{Pattern factor} \qquad f_p = 1 - \frac{1}{J} \sum_{j=0}^{J-1} XOR\left( \sum_{i=0}^{I-1} P_{i,j}, \sum_{i=0}^{I-1} A_{i,j} \right)$$

where $XOR(x, y) := 1$ if $(x \neq 0 \wedge y = 0) \vee (x = 0 \wedge y \neq 0)$, and 0 otherwise.

**KeepTicks Rule.** This rule is a supplement of the KeepInstruments rule. The idea is that patterns, which have strokes on the same tics, have a recognizable musical connection. Therefore, patterns with the same tics as the initial pattern are rated higher than others.

$$\textit{Pattern factor} \qquad f_p = 1 - \frac{1}{I}\sum_{i=0}^{I-1} XOR\left(\sum_{j=0}^{J-1} P_{i,j}, \sum_{j=0}^{J-1} A_{i,j}\right)$$

### 5.2 Reaction Rules

The reaction rules are designed to allow for a musical adaptation of the current solo excerpt to the accompaniment of the virtual drummer. They all have a certain musically meaningful task in the context of the musical dialogue. Therefore, they are only estimated when a certain condition is fulfilled. The fitness of these rules is the arithmetic mean of solo and pattern factor, unless otherwise stated.

**Chromatic Rule.** The smallest movement possible in Western music corresponds to a semitone. If the solo contains many of such chromatic steps, an accompaniment without rests is desirable. The condition to check is that at least two notes are played and that the smallest interval is a minor second. The solo factor is defined as follows. The smaller the average interval is, the greater is the factor's value. The pattern factor increases with the number of *active tics*. A tic is called active, if one cell of the tic has a value greater than the half of the solo's average volume. If a pattern is without a gap, the pattern factor will be 1.

$$\begin{aligned}
&\textit{Condition} && \#_{notes} > 1 \wedge Min_{int} \leq 1 \\
&\textit{Solo factor} && f_s = 1/(MAX\{1, \overline{x}_{int}\}) \\
&\textit{Pattern factor} && f_p = \frac{1}{I}\sum_{i=0}^{I-1} \Theta_1\left(\sum_{j=0}^{J-1} h_{0,5}(P_{i,j})\right)
\end{aligned}$$

where $\Theta_n(x) := 1$ if $x \geq n$, and 0 otherwise. Moreover, $h_n(x) := 1$ if $x > n \cdot \overline{x}_{vol}$, and 0 otherwise.

**FreeJazz Rule.** Common scales in Western music consist of seven pitch classes. If a solo excerpt contains more than seven different pitch classes, uncommon scales are played. This is typical in free jazz. Then, a rather unpredictable accompaniment is desirable. Therefore, the pattern factor is random. The solo factor increases with the number of different pitch classes. Because $\#_{pitches_{12}}$ is maximally 12, the fraction can be maximally equal to $0.8\overline{3}$. Subtracted from 1.08, the solo factor becomes nearly 1.

$$\begin{aligned}
&\textit{Condition} && \#_{pitches_{12}} > 7 \\
&\textit{Solo factor} && f_s = 1.08 - 1/(\#_{pitches_{12}}) \\
&\textit{Pattern factor} && f_s = r \in U[0,1] \quad \text{(uniformly distributed random value)}
\end{aligned}$$

**Holdsworth Rule.** This rule is inspired by the playing style of the British jazz guitar player Allan Holdsworth. He is famous for using large intervals in his solos. This practice is uncommon for ordinary jazz solos and therefore the virtual drummer shall show a special reaction. If the solo excerpt contains intervals larger than 5 semitones in average, the high fitness values are assigned to patterns, which have accents on different instruments for every two consecutive tics. The greater the interval is, the higher the fitness. A tic has an accent if the corresponding cell value is larger than the current solo's average volume.

*Condition* $\#_{notes} > 1 \wedge \overline{x} \geq 5$

*Solo factor* $f_s = MIN\{1, \overline{x}_{int}/12\}$

*Pattern factor* $f_p = \frac{1}{I} \sum_{i=1}^{I-1} 1 - id\left(acc_P(i-1), acc_P(i)\right)$

where $id(x, y) := 1$ if $x = y \geq 0$, and 0 otherwise. The *accent function* is given by

$$acc_P(x) := \begin{cases} J-1 & \text{if } P_{x,J-1} \geq MAX\{P_{x,0}, .., P_{x,J-1}\} \wedge P_{x,J-1} > \overline{x}_{vol} \\ \vdots & \vdots \\ 0 & \text{if } P_{x,0} \geq MAX\{P_{x,0}, .., P_{x,J-1}\} \wedge P_{x,0} > \overline{x}_{vol} \\ -1 & \text{otherwise} \end{cases}$$

**Legato Rule.** Legato indicates that consecutive musical notes are smoothly connected without rests. The accompaniment should avoid rests as well. Legato is detected by the analysis of the gap feature. The smaller the gaps are, the higher is the solo factor. The pattern factor is the same as in the chromatic rule and increases with the number of active tics.

*Condition* $\#_{notes} > 1$

*Solo factor* $f_s = 1/\sqrt{MAX\{1, \overline{x}_{gap}\}}$

*Pattern factor* $f_p$, see Chromatic rule

**Loudness Rule.** If the average volume of a pattern is close to the average volume of the current solo excerpt, the fitness value increases. The fitness value is calculated based on the distance between both average volumes.

*Condition* $\#_{notes} > 0$

*Solo factor* $f_s = \overline{x}_{vol}$

*Pattern factor* $f_p = \frac{1}{\#_{active}} \sum_{i=0}^{I-1} \sum_{j=0}^{J-1} P_{i,j}$

where the *number of active cells of* $P$ is $\#_{active} = \sum_{i=0}^{I-1} \sum_{j=0}^{J-1} \Theta_1(P_{i,j})$.

In this particular case, the fitness rule is $f = 1 - \frac{1}{127}|f_s - f_p|$.

**Ostinato Rule.** An ostinato is a motif or a phrase that persistently repeats. In a solo, the fast repetition of such pattern causes an increase of intensity. This should be maintained in the accompaniment. The condition to check is that in the solo excerpt there must be three times as much notes than different pitches. The solo factor increases with less different pitches and more note repetitions. The pattern factor favors patterns, which have at least two accents at each tic.

*Condition* $\#_{notes} > 0 \wedge (3 \cdot \#_{notes}) \geq \#_{pitches}$

*Solo factor* $f_s = 1 - MAX\{1, \#_{pitches} - 3\} \,/\, \#_{notes}$

*Pattern factor* $$f_p = \left[ \left| I - \sum_{i=0}^{I-1} \Theta_2 \left( \sum_{j=0}^{J-1} h_1 \Big( P_{i,j} \Big) \right) \right| + 1 \right]^{-2}$$

**Pedal Rule.** In tonal music, a pedal note is a long sustained note, typically placed below a melody line. When long notes are played, the drum patterns should contain more rests. The longer a note is, the higher the solo factor will be. For notes longer than 5 s, the solo factor is equal to 1. The pattern factor counts the silent tics (inactive tics) and is equal to 1, when active and inactive tics are at the ratio of 2 to 1.

*Condition* $\#_{notes} > 0$

*Solo factor* $f_s = MIN\{5000, \overline{x}_{leng}\} \,/\, 5000$

*Pattern factor* $f_p = MIN\{1,\ 3 \cdot \#_{inactiveTics} \,/\, (2 \cdot I)\}$

where the number of inactive tics is $\#_{inactiveTics} = I - \sum_{i=0}^{I-1} \Theta_1 \left( \sum_{j=0}^{J-1} P_{i,j} \right)$.

**Staccato Rule.** Staccato is a form of musical articulation with tones of a shortened duration. Hence, staccato is the opposite to legato, and this technique should influence the accompaniment. A note is detected as played with staccato, if its distance to the consecutive note is maximally half of its length. Additionally, it must by shorter than 380 ms. If it is shorter than 80 ms, the solo factor is equal to 1. Patterns with a strong accent every three tics are preferred by the pattern factor. A strong accent is a tic with a cell value equal or greater than 110, corresponding to the intensity of $\frac{110}{127} \approx 86.6\%$.

*Condition* $\#_{notes} > 0 \wedge 2 \cdot \overline{x}_{leng} \leq \overline{x}_{dist} \wedge \overline{x}_{leng} \leq 380$

*Solo factor* $f_s = 1 - MAX(0, \overline{x}_{leng} - 80) \,/\, 380$

*Pattern factor* $$f_p = \left[ \left| \frac{I}{3} - \sum_{i=0}^{I-1} \Theta_1 \left( \sum_{j=0}^{J-1} \Theta_{110} \Big( P_{i,j} \Big) \right) \right| + 1 \right]^{-1}$$

**Virtuoso Rule.** A virtuoso is an individual who possesses outstanding technical skills, in particular, to play very fast. An eligible accompaniment should try to use as many percussive instruments as possible to produce a versatile sound. The more notes are in the solo, the higher is the solo factor. The pattern factor increases with the number of instruments *used*, i.e. instruments which patterns have at least one cell value greater than the average volume of the solo. If all instruments are already used in the initial pattern, this rule does not make sense. Therefore, it is not recommended to use all instruments in the initial pattern.

| | |
|---|---|
| *Condition* | $\#_{notes} > 0$ |
| *Solo factor* | $f_s = (1 - 1 \,/\, \#_{notes})^4$ |
| *Pattern factor* | $f_p = \dfrac{\sum_{j=0}^{J-1} AND\Big(\sum_{i=0}^{I-1} 1 - \Theta_1(A_{i,j}), \sum_{i=0}^{I-1} h_1(P_{i,j})\Big)}{\sum_{j=0}^{J-1} \Theta_1\Big(\sum_{i=0}^{I-1} 1 - \Theta_1(A_{i,j})\Big)}$ |

where $AND(x, y) := 1$ if $x \neq 0 \wedge y \neq 0$, and 0 otherwise.

**Width Rule.** The use of relatively high and low notes in a short period of time causes an increase in intensity, and the accompanying drum patterns should use all instruments to transfer the versatility of sound to the drums. The solo factor is 1, if the range of notes is equal or greater than 36 (3 octaves). The pattern factor increases with the number of non-empty instrument patterns.

| | |
|---|---|
| *Condition* | $\#_{notes} > 0$ |
| *Solo factor* | $f_s = MIN\{36, R_{note}\} \,/\, 36$ |
| *Pattern factor* | $f_p = \dfrac{1}{J} \sum_{j=0}^{J-1} \Theta_1\left(\sum_{i=0}^{I-1} P_{i,j}\right)$ |

### 5.3 Random Rule

The rating is completely random, and the solo excerpt as well as the drum pattern are ignored completely. The random rule can be used to make the virtual drummer more unpredictable by increasing its weighting. This can be interpreted as a creative impulse in a musical dialogue and is responsible for more variation when the population of patterns becomes too uniform.

| | |
|---|---|
| *Fitness* | $f = r \in U[0, 1]$ (uniformly distributed random value) |

## 6 Evaluation

The success of the system can be measured by active and passive test persons (musicians resp. listeners). In particular, it shall be tested to which extent the

**Table 1.** List of all parameters which can be adjusted in real time

| Parameter | Interval | Number range |
|---|---|---|
| Initial pattern | $P_{i,j} \in [0, 127]$ | $\mathbb{N}$ |
| Population size | $N_p \in [1, \infty)$ | $\mathbb{N}$ |
| Sleep time [ms] | $T_z \in [0, \infty)$ | $\mathbb{N}$ |
| Mutation expansion limit | $\mu \in [0, \infty)$ | $\mathbb{N}$ |
| Input window size [ms] | $T_w \in [0, \infty)$ | $\mathbb{N}$ |
| Weight of a rule | $w \in [0, 1]$ | $\mathbb{R}$ |
| Tics-per-minute | $TPM \in [1, \infty)$ | $\mathbb{N}$ |
| Shuffle factor | $f_{shuffle} \in [0, 5, 1]$ | $\mathbb{R}$ |

behavior of the virtual drummer can fit the users' expectations, among others by adjustment of system parameters listed in Table 1.

In a user-related study, active test persons were two pianists, two guitarists, a bassist, and a saxophonist with years of experience in jazz bands and at jazz sessions with real drummers. Only the pianists have the benefit of producing entirely correct MIDI events. The string and reed instruments require audio-to-MIDI conversion. Note that the mistakes made at the MIDI conversion are not critical. They do not lead to a malfunction of the system. We will see that a fully predictable behavior of the virtual drummer is even not desirable. Mistakes can be even interpreted by the soloists as musical ideas developed by the drummer.

The test persons were let to play together with the virtual drummer. Afterwards, points of criticism were discussed. An always mentioned issue was that the reactions of the drummer should be as immediate as possible. This can be achieved by the reduction of $T_z$. For $T_z < 30\,\text{ms}$, its limit could not be observed any more by the calculation. But $T_z = 30$ ms was rated as sufficient enough by all test persons. Additionally, $\mu$ can be increased to produce more versatile offsprings. Therewith, a faster mutation is possible, but the evaluation time increases. To reduce the reaction time, $T_w$ can be manipulated as well. At $100\,\text{ms} < T_w < 1000\,\text{ms}$, the test persons felt a good balance between reaction and detection of solo structures. For $T_w < 100\,\text{ms}$, the system was not able to recognize meaningful solo excerpts. Many excerpts were empty, leading to an undesired relaxation of the accompaniment. Besides, $T_w \geq T_z$ must always hold to not ignore any notes.

The relaxation of the accompaniment ensured by the keep rules when pausing soloing was reported as a particularly positive experience. Even so, the random rule should always be active to preserve the stagnation of accompaniment.

The test persons have expected an autonomic behavior of the virtual drummer which should provide creative impulses and support a musical dialogue with the soloist. To achieve this only by means of the random rule becomes unsatisfying after a while. Another possibility is to increase $N_p$, so that also patterns with lower fitness values are played.

Another point of criticism was the rhythmical limitation of the virtual drummer based on the concept of a drum machine. Sometimes, it appears to be very useful in a musical way to place drum strokes between the tics, for example, to play double time fills. It is not possible to do this dynamically because of the input and mutation concept that bases on the drum machine concept to create instant familiarity while interacting with the system. A possible solution is to double the number of tics and to use only even tics for the definition of the initial pattern. Then, the mutation process will add strokes in between. This is described as musically acceptable but not satisfying in comparison with realistic drum fill playing.

The tests always started with a jazz-typical swing drum pattern in medium tempo with $f_{shuffle} = 0.67$. At higher tempos, $f_{shuffle}$ has to be decreased. For tempos above 250 bpm, it was suggested to set $f_{shuffle} = 0.5$. That was confirmed as a reasonable measure by the test persons. Among other tested pattern styles like rock, funk, hiphop, or latin, a particularly favored pattern was the bossa nova. Because of its driving straight eight groove and its shifting single patterns it was often described as thrilling by the test persons. The accentuated use of toms that seems to be typical for the implemented system was described as working perfectly in this style.

An interesting observation was that the test persons often experienced the limits of the system and explored its possibilities. Uneven time signatures, random or intentionally unconventional weighting, unstructured initial patterns, or unusual drum synthesizer sounds were chosen. This led to very experimental and innovative music that was obviously fun to play. The test persons stated that the overall range of possibilities to adjust the system from producing only small variation of initial pattern to a complete chaotic alienation was very satisfying.

Another attempt was to find musical structures of higher levels in the solos. The virtual drummer failed in this task, because of its limited memory and its lack of planning. The use of a high $T_w$ value led to enormous calculation and impractical reaction times.

All active test persons agreed that the system can improve their way of practicing jazz improvisation. They stated that practicing with a virtual accompanying drummer is more interesting, more exciting, more entertaining, and more versatile. Inactive test persons even experienced the music as concert-like if a virtual drummer was supported with human chord instrumentalist as well as a human bassist additionally.

## 7 Conclusions and Outlook

The aim of this work was to develop a reactive system that is able to create a real-time rhythmical accompaniment for a jazz solo by means of an evolutionary algorithm. The idea derives from the desire to use an interactive playalong track for practising jazz improvisation alone at home. To measure the success of generated drum patterns, a set of evaluation rules was proposed with regard to musical properties of jazz accompaniment.

The system was tested by musicians as well as audience who estimated the musical acceptance of the produced music. It was consistently described as versatile and therefore beneficial for daily practice.

Future work will include improvement of the evaluation rules and their customization. Consideration of weak and strong beats as well as syncopation measuring based on musical theory is planned. Further, it is promising to mix the presented solo and pattern factors to create new rules or to even integrate machine learning approaches like deep neural networks in the rules fitness calculation. Other musically meaningful evolutionary operators can be adopted, e.g., inversions or rotations of patterns. The final attempt will be to create other virtual musicians like pianists, guitarists, or bassists to complete the virtual jazz band and to be able to improvise accompanied by a fully staffed interactive playalong track.

## References

1. Aebersold, J.: Volume 1 - How to Play Jazz & Improvise. Jamey Aebersold Jazz (1967) (Jamey Aebersold Play-A-Long Series)
2. Hughes, C.: Learn Jazz Standards (2010). http://www.learnjazzstandards.com/about/. Accessed 2 Nov 2016
3. Gannon, P.: Band-in-a-Box. PG Music Inc., Hamilton (1990)
4. Biolcati, M.: iReal Pro. Technimo LLC, New York (2008)
5. Biles, J.A.: GenJam: a genetic algorithm for generating jazz solos. In: Proceedings of the International Computer Music Conference (ICMC 1994), San Francisco, USA, pp. 131–137. International Computer Association (1994)
6. Lewis, G.E.: Too many notes: computers, complexity and culture in voyager. Leonardo Music J. **10**, 33–39 (2000)
7. Yee-King, M.J.: The evolving drum machine. In: MusicAL 2007 Proceedings (2007). http://cmr.soc.plymouth.ac.uk/Musical2007/proceedings.htm. Accessed 5 Nov 2016
8. Hoover, A.K., Stanley, K.O.: Exploiting functional relationships in musical composition. Connection Sci. **21**(2–3), 227–251 (2009)
9. Tokui, N., Iba, H.: Music Composition with Interactive Evolutionary Computation. Graduate School of Engineering, The University of Tokyo (2001). http://www.generativeart.com/on/cic/2000/ga2000-tokui.htm. Accessed 29 Oct 2016
10. Unemi, T., Nakada, E.: A tool for composing short music pieces by means of breeding. In: Proceedings of the 2001 IEEE Systems, Man and Cybernetics Conference, pp. 3458–3463 (2001)
11. Dostál, M.: Genetic algorithms as a model of musical creativity - on generating of a human-like rhythmic accompaniment. Comput. Inform. **22**, 321–340 (2005)
12. MMA, MIDI Manufacturers Association: General MIDI 1, 2 and Lite Specifications (1991). https://www.midi.org/specifications/category/gm-specifications. Accessed 6 Nov 2016