

An Intelligent Interface for Drum Pattern Variation and Comparative Evaluation of Algorithms

RICHARD VOGL¹, MATTHIAS LEIMEISTER², CARTHACH Ó NUANAIN³,
(richard.vogl@jku.at) (matthias.leimeister@native-instruments.de) (carthach.onuanain@upf.edu)

SERGI JORDA³, MICHAEL HLATKY², AES Member, AND PETER KNEES¹

¹*Department of Computational Perception, Johannes Kepler University Linz, Austria*

²*Native Instruments GmbH, Berlin, Germany*

³*Music Technology Group, Universitat Pompeu Fabra, Barcelona, Spain*

Drum tracks of electronic dance music pieces are a central and style-defining element. Yet, creating them can be a cumbersome task, mostly due to lack of appropriate tools and input devices. In this work we use a UI prototype that aims at supporting musicians to compose the rhythmic patterns for drum tracks, to compare different algorithms for drum pattern variation. Starting with a basic pattern (seed pattern), which is provided by the user, a list of variations with varying degrees of similarity to the seed pattern is generated. The variations are created using one of the three algorithms compared: (i) a similarity-based lookup method using a rhythm pattern database, (ii) a generative approach based on a stochastic neural network, and (iii) a genetic algorithm using similarity measures as a target function. The interface visualizes the patterns and provides an intuitive way to browse through them. User test sessions with experts in electronic music production were conducted to evaluate aspects of the prototype and algorithms. Additionally a web-based survey was performed to assess perceptual properties of the variations in comparison to baseline patterns created by a human expert. The web survey shows that the algorithms produce musical and interesting variations and that the different algorithms have their strengths in different areas. These findings are further supported by the results of the expert interviews.

0 INTRODUCTION

Nowadays, more than ever before, digital tools for music production play an important role in the workflow of music producers. Such tools cover applications like digital audio workstations (DAW), integrated hardware/software solutions like grooveboxes, and software tools and plugins like synthesizers and audio effects. In the *GiantSteps* project, we focus on simplifying the workflow of music producers by developing *intelligent agents* for the usage in electronic dance music (EDM) production and performance.¹

Usually, drum tracks are built by arranging rhythm patterns from a pattern library or by creating patterns manually. Using predefined patterns bears the risk of sounding unoriginal, while creating them manually is a time-consuming task and requires more musical knowledge. Entering rhythm patterns in a DAW is typically done using a mouse or MIDI controllers (keyboards and drum pads) to set notes in a piano roll or similar editor. Step-sequencer-like interfaces are

usually a feature of grooveboxes and drum machines and are found in many setups for live performances.

The aim of this work is to build a tool that supports musicians in an intuitive way at creating variations or finding inspiration for new drum rhythm patterns. Maintaining the basic style, or rhythmical idea, while providing meaningful and interesting—maybe even surprising—variations is a main goal.

When it comes to samplers and synthesizers for drums in EDM, a wide variety of commercial products as well as a lively research community exist. However, there are few works on automated drum rhythm variation and creation.

1 RELATED WORK

Some commercial products in the field of music production include tools that attempt to automate the creation of a drum track. In Apple's Logic Pro software,² the user can

¹ <http://www.giantsteps-project.eu/>

² <http://www.apple.com/logic-pro/>

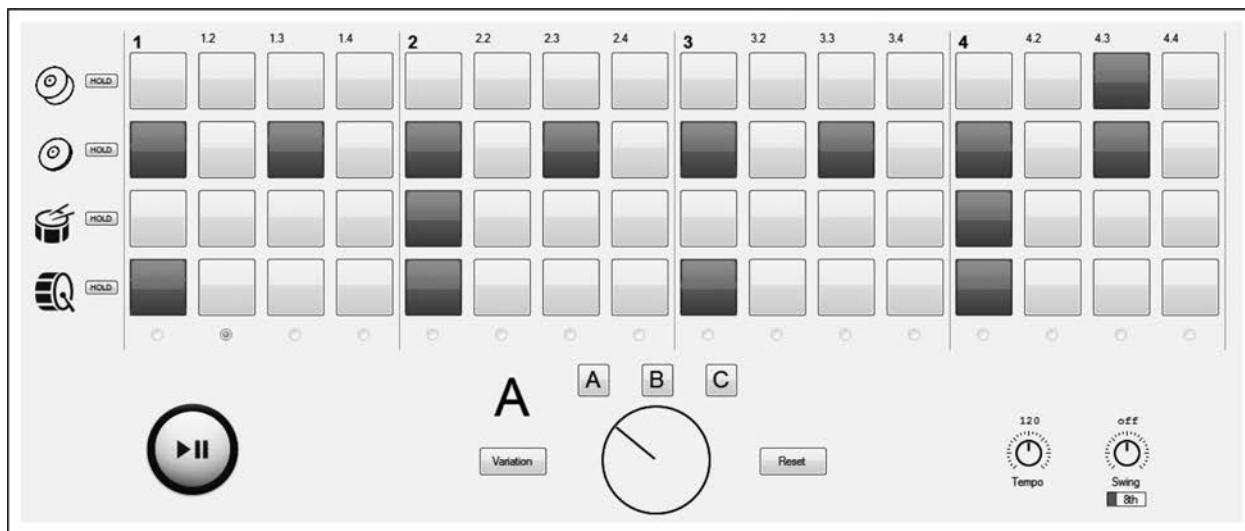


Fig. 1 Screenshot of the prototype. The 4-by-16 step sequencer array acts as visualization and input for the drum rhythm patterns. Beneath the array are controls for playback, the pattern variation control, buttons for algorithm selection, and controls for tempo and swing (all also controllable through a hardware interface).

create a *Drummer* track that contains basic drum patterns that can be changed via an interactive GUI. After choosing a genre-specific drummer (e.g., Rock) and drum kit, the style of the drum track can be varied by moving in a two-dimensional panel that represents the dimensions soft-loud and simple-complex. Steinberg's Groove Agent³ is a drum plugin that comes with a big selection of drum kits and loops that can be arranged to a full drum track in a DAW project. It includes a mode for performance (*Style Player*), in which the user can choose different styles of patterns based on intensity and complexity. An interesting feature is the automatic mode that varies the complexity of patterns on the fly. EZ Drummer by Toontrack⁴ provides a large library of drum patterns together with the option for searching matching patterns based on a seed pattern that the user can record. Since these products are mainly positioned for the production of rock and alternative music, they are only to a certain degree applicable for EDM. Furthermore, professional producers often refrain from using out-of-the-box patterns for fear of sounding unoriginal.

The scientific community mainly focuses on methods and algorithms providing the underlying functionality. Of special interest is the question how the human perception of rhythm, especially the notion of similarity, can be modeled. Toussaint [20] introduces and compares several measures for the similarity of two rhythmic patterns. Among them are the Hamming distance, edit distance, Euclidean distance of inter-onset-interval vectors, and the interval-ratio-distance. To gain insight into similarity of the patterns, phylogenetic trees are built based on the computed distance matrices—a bioinformatics technique that is originally used to visualize the relationship of DNA sequences. In the work of Kaliakatsos-Papakostas et al. [9] an automatic drum rhythm generation tool based on genetic algorithms is described. It

generates variations of a base rhythm and allows the user to set parameters of the generation process such as divergence between the base rhythm and the generated ones. Ó Nuanáin et al. [12] use a similar approach for rhythm pattern variation using genetic algorithms. Sioros and Guedes [15] introduce a system that recombines MIDI drum loops in realtime based on a measure for complexity of rhythmic patterns. The complexity is measured by means of syncopation and density. A more detailed discussion of different approaches for measuring syncopation in the same context is presented in [16]. Apart from approaches that compute similarity between rhythmic patterns on a symbolic representation, there exist methods that consider other aspects of rhythmic similarity. Holzapfel and Stylianou [7] use the scale transform to develop a tempo invariant rhythmic similarity measure for music. Jensen et al. [8] as well as Gruhne and Dittmar [5] use logarithmic autocorrelation functions calculated on different forms of onset density functions to obtain tempo invariant rhythmic features. Other works investigate properties of swing and try to quantify the swing-ratios of rhythmic patterns in audio recordings, e.g., [3, 11].

Given training data, machine learning methods can be used to train generative models for rhythm patterns. Paiement et al. [13] introduce a probabilistic model for relative distances between rhythms, which is applied to subsequences of patterns. This is in turn used to predict the continuation of a rhythmic pattern given its past, utilizing a hidden Markov model (HMM). Another group of widely used generative models are restricted Boltzmann machines (RBM) [6]. Boulanger-Lewandowski et al. [2] use an extension of RBMs with recurrent connections to model and generate polyphonic music. Battenberg and Wessel [1] use another variant, the conditional RBM, to analyze drum patterns for their meter. They mention the capability of the learned model to generate drum patterns similar to the training data given a seed pattern. This idea was further

³ http://www.steinberg.net/en/products/vst/groove_agent/

⁴ <https://www.toontrack.com/product/ezdrummer-2/>

developed in the work by Vogl and Knees [21] to build a drum rhythm variation tool based on an RBM and a step sequencer interface. In this work, this prototype is further extended by incorporating and extensively comparing two more variation algorithms.

2 PATTERN VARIATION USER INTERFACE

To be able to compare and evaluate different algorithms, we extended the interface prototype presented in [21] to accommodate three variation methods that can be selected via the UI. Fig. 1 shows the UI of the prototype.

It resembles a standard drum step sequencer, providing four instruments (bass-drum, snare, open, and closed hi-hat) programmed within one 4/4 measure bar of 16th notes. This type of interface was chosen since it represents one of the prevalent interfaces for drum track creation in EDM. In this context, we focus solely on the variation of the plain symbolic rhythm patterns. Hence, we deliberately decouple the pattern from aspects of accentuation, micro timing, and swing, which are considered independent dimensions and should remain under the control of the artist. For controlling tempo and swing, the presented UI exhibits respective knobs. The latter allows the user to set an additional swing ratio that shifts either 8th or 16th notes to create a “swing feel.” These parameters enable the users to recreate a rhythmic style they usually work with, while keeping the UI complexity at the necessary minimum.

A central UI element is the variation browsing knob and the controls to set and reset the pattern for which the variations should be created. After pressing the “variation” button, the selected algorithm generates variations. These variations are ordered ascending regarding the number of active notes. By turning the variation knob to the left, the variations will become more sparse; when turning the knob to the right the variations will become more dense. By pressing the reset button, the variation knob jumps back to the seed pattern. Above the variation knob are buttons to select one of the three variation algorithms. The assignment is randomized after every start of the prototype to avoid any bias introduced by the order. The output of the interface is sent via MIDI. All UI elements can be controlled via MIDI, making it possible to use an external hardware controller, or integrating the prototype into a DAW software.

3 PATTERN VARIATION ALGORITHMS

The aim of the presented algorithms is to support the user in creating interesting drum tracks based on an initial seed pattern. Hence, the variation algorithm receives a one-bar pattern of kick, snare, closed hi-hat, and open hi-hat notes as input. Based on this, a set of 32 new patterns is computed and returned to the UI as suggested variations of the seed pattern. This set is sorted according to density of note events per bar. The seed pattern is placed in the list according to its own density as a starting point for exploring the patterns. Two of the following algorithms are data-driven as they require model training or a database lookup. Therefore, we first describe the collection of rhythm patterns that has

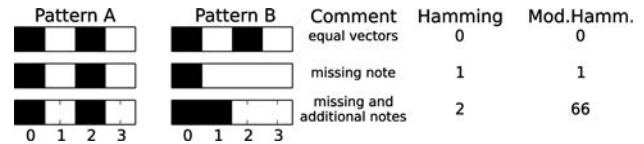


Fig. 2. Rhythm patterns as binary vectors of size four. Depicted are examples for two equal vectors, vectors with missing notes (or additional notes if the right vector is used as reference), and vectors with both missing and additional notes. The columns to the right show the corresponding Hamming and modified Hamming distance.

been used throughout the experiments as database and for model training.

3.1 Data Set

Maschine⁵ is an integrated hardware/software groove box application that is focused on the production of urban and electronic music. The application and its extension packs contain a large library of drum sound samples and examples for drum patterns.

This collection of drum patterns was used to compile a data set for the development of rhythm pattern variation algorithms. Using the MIDI export function of the application, the example drum patterns have been exported to MIDI files and cut into patterns of one bar length. From the exported samples, only the instruments kick, snare, closed hi-hat, and open hi-hat were used. The resulting patterns were checked for exact repetitions and some were removed based on musical constraints. For example, only patterns containing between two and six bass drum notes per bar, between one and five snare drum notes, as well as at least two hi-hat notes were kept. This was done to exempt breaks and very sparse patterns from the database. The final set consists of 2,752 unique patterns.

3.2 Modified Hamming Distance

Both the database-driven approach (Sec. 3.3) as well as the neural-network-based method (Sec. 3.4) use a rhythm pattern similarity function. To calculate the similarity, the rhythm patterns are represented as 64-bit (16 notes, 4 instruments = 64 notes) binary vectors. Then the number of bits with different values between the two vectors is counted (= Hamming distance). An additional offset (64) is added if one vector compared to the other has additional as well as missing notes. See Fig. 2 for examples. This choice was made based on the findings in [20] and [12] as well as the goal to favor patterns that have as much overlap as possible with the seed pattern. It is supposed to create the sensation of a steady evolution when browsing the resulting list of variation patterns, which was sorted using this distance function.

3.3 Database-Driven Approach

The database-driven algorithm (referred to as *DB* in the following) is based on retrieving patterns similar to the seed

⁵ <http://www.native-instruments.com/products/maschine>

pattern from a database and suggesting these as variations. For the rhythm pattern database, the patterns extracted from Maschine (cf., Sec. 3.1) were used. The modified Hamming distance is used to compute the similarity of the seed pattern (query) to every pattern in the database (targets). In case the target pattern contains more notes, but also misses notes from the query, the pattern is modified by adding the missing notes from the query pattern. As a result, generated patterns with more notes than the query always contain all notes from the query pattern. The resulting patterns are sorted according to the computed distance and a list of the 32 most similar patterns is returned.

3.4 Neural Network Based Method

To generate meaningful, yet creative patterns, a process that combines obedience to musical rules with elements of unpredictability is needed. To achieve this, Restricted Boltzmann machines (RBMs, introduced in [17]), which are generative stochastic neural networks, are used. From training, they learn a probability distribution defined by the training data. From this distribution samples can be drawn, which can be used for pattern generation.

RBMs consist of two layers of artificial neurons: the visible layer, which is used as both in- and output and a hidden layer that represents latent variables. The neurons are fully linked only between the two layers (hence the name “restricted”). Fig. 4 depicts an example of a small RBM with four visible nodes and three hidden nodes. The RBM used in this work consists of 64 nodes in the visible layer, which correspond to the notes (16x4) in the step sequencer grid (see Fig. 1). The hidden layer consists of 500 nodes. The training was done using the Lrn2 framework of the Lrn2Cre8 project.⁶

The RBM is trained using the Maschine rhythm pattern data set by means of persistent contrastive divergence (PCD) [19] training. Additionally, latent selectivity and sparsity, as described in [4], as well as drop-out [18] for training are used.

To generate variations of the seed pattern, first, the seed pattern is entered into the visible layer of the RBM. Then, variations for every instrument are generated individually by clamping (nodes fixed to their original input values) all other instruments and performing several Gibbs sampling steps. A Gibbs sampling step consists of (i) calculating the values for the hidden layer by using the input values in the visible layer and the learned network weights, (ii) binarizing the values of the hidden layer by applying the sigmoid function and a random threshold, and (iii) calculating new values for the visible layer by using the values in the hidden layer and the network weights. Fig. 3 shows the evolution of the visible layer of the RBM performing Gibbs sampling steps. It can be observed how the snare pattern (nodes 16–31) evolves while the other instruments (nodes 0–15 and 32–63) are clamped to their original values. For more details on Gibbs sampling, clamping, and RBM training, the reader

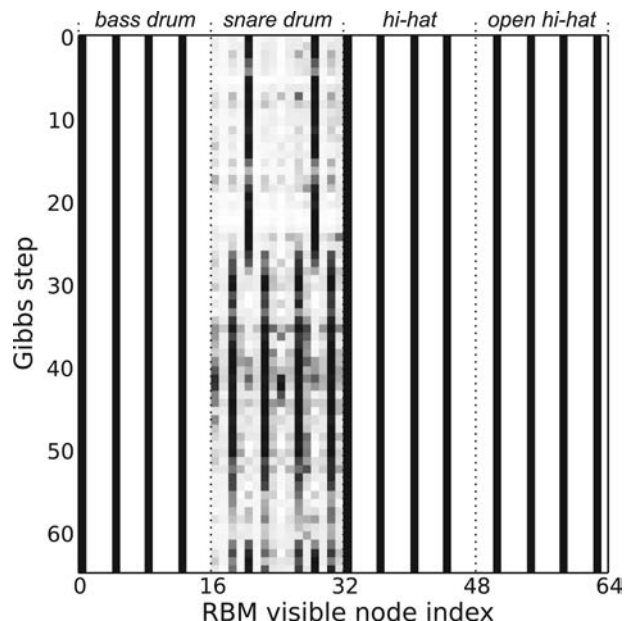


Fig. 3. The evolution of the visible nodes of the RBM while creating pattern variations for the snare drum. The x-axis represent the index of the visible node of the RBM. The y-axis represents the index of the Gibbs sampling step, starting at the top with the original input pattern and progressing downwards. Active nodes are represented by black, inactive nodes by white pixels.

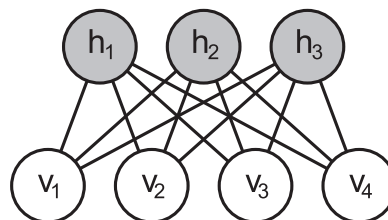


Fig. 4. Structure of a Restricted Boltzmann Machine. White circles represent the visible and gray circle the hidden nodes. Between the two layers, the nodes are fully connected, while nodes within the same layers are not connected.

is referred to the work by Hinton et al. [6] and the other references provided.

The generated single-instrument patterns are sorted using our modified Hamming distance. The sorted single-instrument pattern lists are then combined to full rhythm patterns by using bass drum, snare drum, open, and closed hi-hat patterns at the same indices. This approach will subsequently be referred to as *RBM*.

3.5 Genetic Algorithm

Genetic algorithms are frequently used in algorithmic composition and other creative applications in what is often called “generative art.” In essence, a genetic algorithm involves successive splicing, pairing, and mutating of data structures in a simulated program of natural selection. At the core of a genetic algorithm lies the fitness function that determines whether individuals fulfill some solution criteria.

⁶ <https://github.com/OFAI/lrn2>

Table 1. Mnemonics used in the article.

Algorithms	<i>DB</i>	database based method
	<i>RBM</i>	neural network based method
	<i>GEN</i>	genetic algorithm based method
	<i>EXP</i>	patterns created by an expert
Variations	<i>sp1</i>	sparse variation close to seed (−3)
	<i>den1</i>	dense variation close to seed (+3)
	<i>den2</i>	dense variation far from seed (+6)

In [12], a genetic algorithm that generates rhythmic patterns using an automatic fitness function by determining similarity of new patterns to a seed pattern, was presented. In this work an adapted version of this method is used to meet the constraints and requirements of the UI prototype.

The initial population pool is initialized with a uniformly random distributed set of 16x4 binary pattern genomes, conforming to the kick, snare, and hi-hat representation used in this work. The algorithm commences and iterates through successive stages of evolution. During a single stage of evolution, patterns from the population pool are selected and paired for mating. New patterns are generated by selecting two parent patterns for crossover (bits from each are copied to the child) and mutation (a very small portion of the child is randomly modified). The fitness of these new patterns is determined by measuring their rhythmic similarity to the input target pattern using a distance function. The evaluation in [12] revealed that the Hamming distance correlates best with human judgments when dealing with polyphonic patterns. For this reason, as well as to remain consistent with the distance measures used in the other algorithmic approaches, in this work the Hamming distance is used.

Achieving a balance between pattern diversity and reasonable convergence time is one of the main challenges. To this end, two adjustments to the method presented in [12] are made. First, a more conventional roulette selection scheme [14] is adapted: candidates from the population are chosen for pairing and splicing from the fittest 100 members only. Second, a stage of elitism in the selection procedure is introduced: a small fraction of the fittest individuals are simply copied to the next generation. This fraction can be tweaked for drastic decreases in the algorithm's convergence time.

We run the algorithm and store the best member from each generation in a list to get a diverse spread of rhythmic patterns in terms of target similarity. From this list 32 patterns are chosen from equidistantly distributed indexes, to be returned to the UI. The genetic algorithm approach will subsequently be referred to as *GEN*.

Table 1 summarizes the mnemonics of the different algorithms for better comprehension.

4 EVALUATION—EXPERT INTERVIEWS

To evaluate all three algorithms as well as the UI prototype, two separate studies have been carried out. The first study consists of interviews conducted with experts in EDM production and performance. The study's aim was to gather

feedback on the quality of the variations produced by the single algorithms as well as on the usability of the UI. Sec. 5 covers the second study. It is a web-based survey in which variations of selected seed patterns were to be rated. The goal of this survey was to obtain quantitative data on various properties of the variation algorithms.

4.1 Method

We conducted interviews with musicians experienced in working with DAWs and producing EDM and/or performing EDM live. The interviews were conducted in a guided and informal way inspired by the “thinking aloud” method [10]. The participants were introduced to the prototype by briefly explaining aim and functionality. Then they were asked to explore the functions and algorithms of the prototype while talking about their experience and thoughts. For every interview, the algorithm button assignment was randomized to avoid experimenter bias. The interviews were recorded and transcribed for later analysis.

The aim was to get answers to five core aspects we deemed crucial for the success of a rhythm pattern variation tool:

1. Is the basic rhythm of the seed pattern preserved in the variations?
2. Are variations musically meaningful?
3. Is the interaction with the prototype intuitive?
4. Would the prototype be useful in a live environment?
5. Would the prototype be useful in a studio/production environment?

Additional comments on feature requests, use-case scenarios, and UI interaction were collected.

4.2 Interview Results

In total, 11 interviews were conducted with musicians in their private studios, at the HAMR'15 hackday in Málaga, and at the Red Bull Music Academy in Paris. After being introduced to the prototype, the participants spent on average 23 minutes (min.: 17, max.: 30 minutes) exploring the behavior of the three pattern variation algorithms (i.e., about 8 minutes per algorithm on average). Five participants used multiple seed patterns (up to three patterns), while the others just used one seed pattern to generate variations. Participants were encouraged to browse through the whole list of variations to get an overall image of how the single algorithms behave. Table 2 summarizes the comments of the interviewees regarding the five aspects of interest identified earlier.

Comments were considered to be positive whenever the interviewees explicitly expressed a positive impression regarding the aspects. For example:

- Rhythm and musicality:
“That [generating variations] actually worked: It adds stuff on the right, it removes on the left.” JKU-15-09

Table 2. Number of participants giving positive responses *wrt.* the topics of interest of the user study.

Aspect	Algo.	Positive comments
rhythm is preserved	DB	10 91.0%
	RBM	8 72.7%
	GEN	5 45.5%
patterns are musical	DB	11 100%
	RBM	10 91.0%
	GEN	7 63.6%
prototype interaction		9 81.8%
would use live		6 54.5%
would use in studio		10 91.0%

Note: The total number of participants (N) was eleven.

- Prototype interaction:
“It works like it should, so I think it is quite user friendly. [. . .] I also think the scrolling [through the variations] is cool because it is fast and practical.” JKU-15-05
- Studio and live usage:
“I would say it would be interesting, in this form, for a studio [. . .]. But it would be very inefficient in a live setting.” RBMA-15-04

Most feature requests were uttered in the context of live environments. The most demanded features requested were, among other things: (i) a preview function, visually or audible—mentioned six times, (ii) a way to store or bookmark patterns—mentioned four times, and (iii) an option to make patterns switch only on the downbeat during playback—mentioned twice.

5 EVALUATION—WEB SURVEY

To substantiate the findings from the expert interviews and evaluate properties of the generated rhythm patterns quantitatively, we additionally conducted a web-based survey. Based on the feedback gathered so far, we were specifically interested in the algorithms’ generated variations in terms of rhythm preservation, difference in detail, interestingness, suitability as substitution, and suitability as fill (see below). We define four research questions (RQ) covering these properties. These RQs are additionally used to structure the evaluation and results (cf., Sec. 5.2):

- RQ I: Are there significant differences between the individual algorithms for each property?
- RQ II: Do the variations capture the basic rhythm of the seed patterns?
- RQ III: Is the sorting within the list provided by the algorithm reasonable?
- RQ IV: Are the investigated properties independent or are these aspects correlated?

These properties are evaluated against a baseline consisting of pattern variations created by a human expert.

Table 3. List of seed patterns of the web survey.

Pattern Name	Pattern Structure
Four To The Floor <i>120 bpm</i>	
Drum And Bass <i>160-180 bpm</i>	
House <i>120-130 bpm</i>	
Techno <i>120-130 bpm</i>	
Reggaeton <i>90-130 bpm</i>	
Funk <i>80-100 bpm</i>	
Hip Hop <i>80-100 bpm</i>	
Dubstep <i>70 bpm</i>	

Note: OHH stands for open hi-hat, HH for hi-hat, SD for snare drum, and BD for bass drum. Beneath the pattern name a typical tempo range for the pattern is provided. As tempo for the audio renderings, the mean of the range was chosen.

5.1 Method

Eight one-bar rhythm patterns (see Table 3), which represent distinctive characteristic styles found in electronic and urban music, were selected as basic seed patterns. For each basic pattern/algorithm combination, three variations were generated. Additionally, posing as a fourth “algorithm” and providing a baseline, a human, i.e., a musician familiar with different EDM styles, created three variations for each seed pattern (referred to as *EXP* in the following). The three variations were taken from the variation lists generated by the algorithms in an automated process. They were selected at the following distances to the seed within the list: -3 (sparse variation; in the following referred to as *sp1*), $+3$ (dense variation; *den1*), and $+6$ (*den2*). The mnemonics for the variations can also be found in Table 1. This results in a total number of 12 variations (four algorithms, each three variations) for one seed pattern. The maximum total number of patterns to evaluate per survey participant was, therefore, 96 patterns.

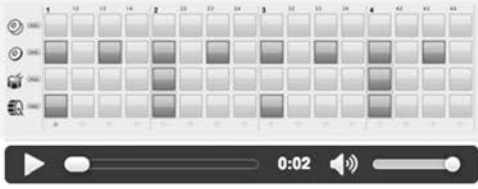
In the survey, we asked five questions to investigate the research questions defined earlier:

- How well does this pattern capture the basic rhythm of the original pattern? (Referred to as *rhythm* in the following)

GiantSteps Rhythm Pattern Survey

Please listen to the original pattern and read the description for the properties we interested in on the right. Listen to the other patterns below and compare them to the original pattern to provide your evaluation of the properties on the scale. Try to use the full scale of options for your answers!

Original Pattern



basic rhythm: How well does this pattern capture the basic rhythm of the original pattern? (6 = Very good)

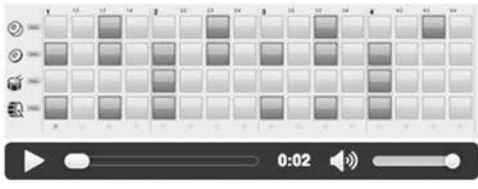
difference: How different in terms of details is this pattern compared to the original pattern? (6 = Very different)

interesting: Consider this pattern a variation of the original. How interesting is this pattern as a variation of the original pattern? (6 = Very interesting)

substitution: Would it make sense to use this pattern as a substitute for the original pattern as a continuous rhythm pattern in a song? (6 = Totally would make sense)

fill: Would it make sense to use this pattern as a single bar variation (drum-fill) of the original pattern within a song? (6 = Totally would make sense)

Pattern 1



basic rhythm	1	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	6
difference	1	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	6
interesting	1	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	6
substitution	1	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	6
fill	1	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	6

Fig. 5. Evaluation page of the web survey. On the left side the original pattern, and the first variation can be seen. In the top right, the single questions are explained. In the lower right the six-point Likert scales for the five questions are shown.

- How different in terms of details is this pattern compared to the original pattern? (*difference*)
- Consider this pattern a variation of the original. How interesting is this pattern as a variation of the original pattern? (*interesting*)
- Would it make sense to use this pattern as a substitute for the original pattern as a continuous rhythm pattern in a song? (*substitution*)
- Would it make sense to use this pattern as a single bar variation (drum-fill) of the original pattern within a song? (*fill*)

The answers to these questions were collected as the score on a six-point Likert scale (no neutral answer, thus “forced choice”).

The survey was conducted by means of a web application. On the first page general information such as age, gender, and questions about musical activity were collected. Subsequently, every seed pattern was represented on a single page where answers to the five questions for the twelve variations had to be provided. Visual aids in the form of images of the rhythm pattern in the step sequencer grid, as well as a rendered audio version of the rhythm patterns using genre specific drum kits were provided. Fig. 5 shows a screenshot of a evaluation page of the web survey. The sequence of the seed patterns (= survey pages) as well as the sequence of the variations was randomized per user to avoid bias caused by a certain order of the variations. It was not mandatory that all seed patterns were processed by every participant. The progress was stored after each page and it was possible to continue the survey at another point in time.

5.2 Survey Results

The web survey was online from October 2015 until January 2016 and was distributed to local contacts and advertised on the Music-IR and the SMC mailing lists. In total, 43 users participated for which 1,536 entries were recorded. Every entry consists of scores for the five survey questions (*rhythm*, *difference*, *interesting*, *substitution*, and *fill*) for one of the three variations (*sp1*, *den1*, and *den2*) of a certain algorithm (one of *DB*, *RBM*, *GEN*, or *EXP*) of a specific seed pattern. This results in 384 data points per algorithm and 512 data points per variation. Fig. 6 shows the distribution of data points among the single seed patterns.

The resulting data set has a strong bias towards male participants: Only 2 out of 43 participants (4.7%) are female. Over 90% (39 out of 43) of the participants are able to read sheet music and almost half of them (20 out of 43) play some kind of percussive instrument.

We conducted one-way ANOVA analysis with consecutive Post-Hoc tests to analyze if the mean scores of the answers to the individual survey questions have significant differences, i.e., we test against the null hypothesis that there is no difference in the mean scores of the different algorithms (including the expert user *EXP*). First, the single distributions were tested for homogeneity of variances using Levene’s test. In case of homogeneous variances an ANOVA with subsequent Ryan-Einot-Gabriel-Welch Range Post-Hoc test was used. In case of inhomogeneous variances, additionally a Welch test with subsequent Games-Howell Post-Hoc test was used ($p = 0.05$ for all tests). This setup is used to find answers to the four research questions (RQ) defined earlier:

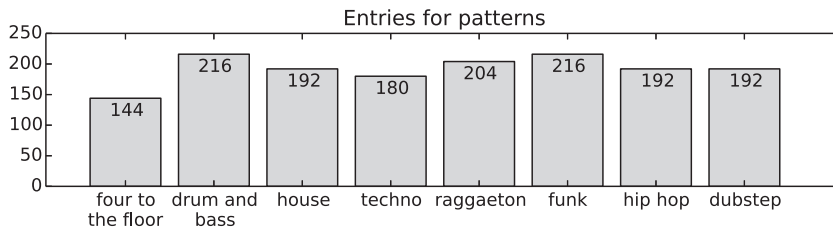


Fig. 6. Distribution of data points among the single seed patterns used in the web-based survey.

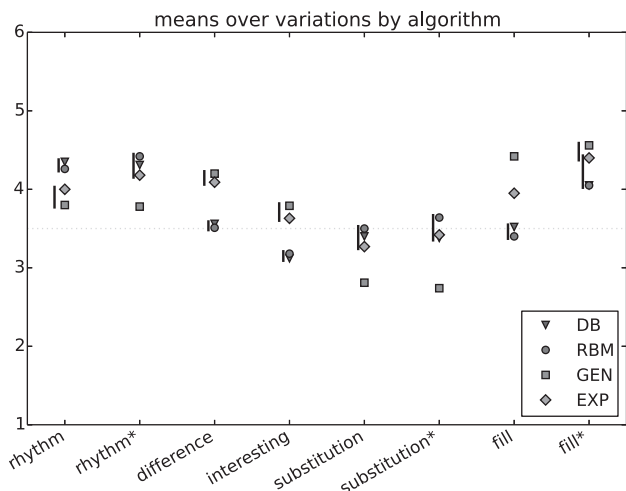


Fig. 7. Mean scores of each algorithm for the aspects evaluated. The black bars indicate homogeneous subsets according to significance analysis. The series with asterisk use only a subset of the full data set: *rhythm** and *substitution** are calculated using only variations *sp1* and *den1*, *fill** was calculated using only variation *den2*.

RQ I: Are there significant differences between the individual algorithms for each property? Fig. 7 shows the evaluation results comparing the means of the individual algorithms for all survey questions. The baseline is represented by the scores for the expert patterns (*EXP*) that are visualized as diamonds in Fig. 7.

For this evaluation, apart from the full data set, two data subsets were additionally used: In the case of *rhythm** and *substitution** all entries with *den2* as variation were excluded. This was done to check if scores for *rhythm* and *substitution* increase if only the similar variations are considered. In the case of *fill** only entries with the more distant variation *den2* were used. This set was used to gain insight if scores for *fill* increase for dense patterns which are more different.

For *rhythm*, all algorithms perform equally well or better than the baseline. Significantly worse than the baseline are: *GEN* in the case of *rhythm**, *DB* and *RBM* in both cases of *difference* and *interesting*, *GEN* in both cases of *substitution* and *substitution**, and *DB* and *RBM* in the case of *fill*. In all other cases the algorithms perform equally well or better than the baseline patterns created by the expert.

The algorithms are able to reproduce the basic rhythm pattern of the seed patterns as good as the expert. While *RBM* and *DB* fail to produce as interesting and differ-

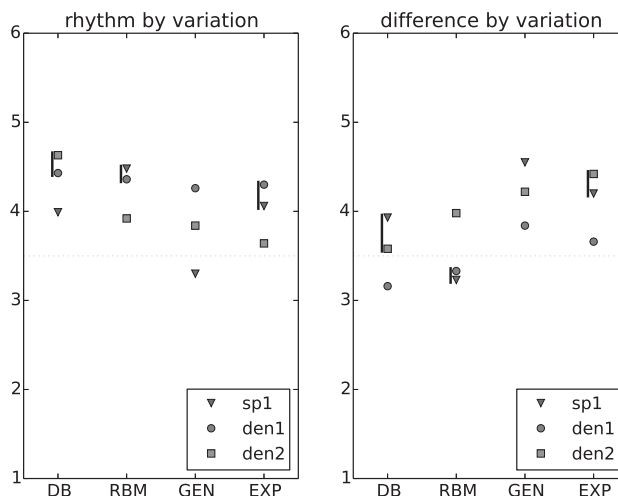


Fig. 8 Mean scores for rhythm (left) and difference (right) for each algorithm split by variation. The black bars indicate homogeneous subsets according to significance analysis.

ent patterns as the expert, *GEN* does not generate substitutes as well as the expert. Since *fill** is more relevant in a real-live scenario (in the interviews users tended to browse to the far right to look for fills), all algorithms can be considered to produce fills equally well as the expert, while *GEN* still performs significantly better than *DB* and *RBM*.

RQ II: Do the variations capture the basic rhythm of the seed patterns? The analysis to *RQ I* shows that for *rhythm*, all algorithms perform equally well or better than the human expert when calculating means over the full data set. For this question the data is split into three subsets for the variations *sp1*, *den1*, and *den2*. This is done in order to check if *rhythm* stays the same regardless of the degree of variation. In the left plot of Fig. 8 it can be observed that *rhythm* does not stay the same for the single variations, not even for *EXP* which was the baseline. In the case of *RBM* and *EXP* only the pattern *den2* scores significantly worse. *DB* performs equally well for *den1* and *den2* but significantly worse for *sp1*. Also *GEN* scores worse for *den2* and even more so for *sp1*.

We can summarize that for *den2* the *rhythm* scores are generally worse than for *den1*. This leads to the assumption that it is difficult for the algorithms (but also for the expert) to reproduce the basic rhythm for more different patterns. The only exception to this is the *DB* approach, which can be explained by the fact that *DB* never changes the basic rhythm pattern when producing patterns with more notes,

as explained in Sec. 3.3. On the other hand *DB* seems to fail to provide sparse patterns with the same basic rhythmic structure. This is probably caused by the fact that the database consists of a limited number of patterns, therefore it is hard to find sparse patterns with the same basic rhythm. *GEN* fails to provide good patterns not only for *den2* but also for *sp1*, which was also observable in the interviews:

“To the left it [GEN] just goes crazy.” JKU-15-01

“If I turn to the right, there were many things which were OK, but on the left side not so much. [...] [GEN] went completely crazy on the left side.” JKU-15-03

While the results for *RBM* are comparable to the expert baseline (*EXP*), *GEN* and *DB* seem to have problems reproducing sparse patterns that capture the basic rhythm.

RQ III: Is the sorting within the list provided by the algorithm reasonable? To evaluate this question two assumptions regarding the scores for *difference* are tested. The first assumption is, that variations *sp1* and *den1* score equally since they were taken at the same distance to the seed pattern. Second, for variation *den2* the score should raise since the distance was twice as large compared to *den1*. The right plot in Fig. 8 shows the evaluation results for *RQ III*. Only for *RBM* both assumptions are true. The second assumption is true for all algorithms, only the assumption that *sp1* and *den1* score equally has to be rejected for *DB*, *GEN*, and *EXP*. *sp1* is, in fact, always rated to be more different than *den1*. Since this is even the case for the baseline (*EXP*) one could assume that sparse patterns are generally perceived more different than dense patterns with the same distance.

Apart from the fact that sparse patterns generally seem to be rated more different than dense patterns, the sorting can be considered reasonable.

RQ IV: Are the investigated properties independent or are these aspects correlated? Fig. 9 visualizes the correlation between the answers to the survey questions. Only the pair *rhythm/fill* shows no significant correlation, which might imply that for fills it is not important if the basic rhythmic feel is preserved. *Substitution* and *fill* show a weak negative correlation that seems reasonable, since patterns rarely qualify for both categories. *Interesting* shows a slight positive correlation with *rhythm* and *difference*. This could imply that participants only find patterns interesting if they conserve the basic rhythm while introducing change. There is a strong positive correlation between *interesting* and *fill* that implies that participants tend to consider interesting patterns suitable as fills. *Difference* shows a weak positive correlation with *fill* that might imply that fills are supposed to be different from the basic rhythm. *Substitution* and *rhythm* also turn out to correlate strongly positively, which comes as no surprise since substitutes should, in general, be similar to the basic rhythm patterns. The same line of reasoning can be applied to the strong negative correlation between *substitution* and *difference*. Since *rhythm* also correlates strongly negatively with *difference*, the aforementioned correlation might be merely a transient

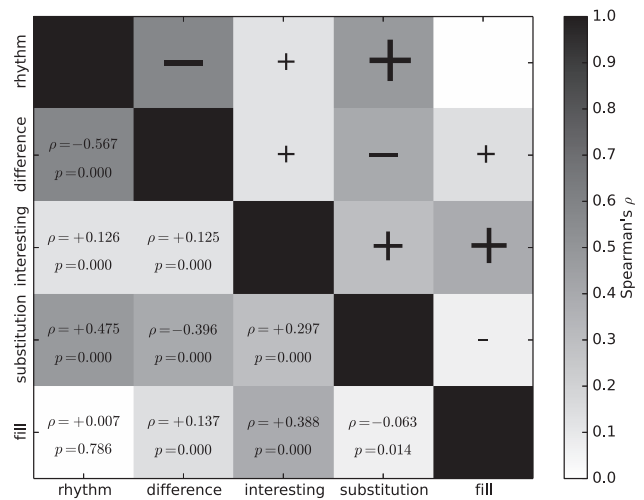


Fig. 9. Symmetric correlation matrix of Spearman's correlation values (ρ) and significance levels (p) for the answers to the survey questions. The upper right half visualizes the value (darker=higher) and direction (negative / positive) of the correlation. The lower left half contains the numeric values.

effect caused by the two very strong correlations of *rhythm* with *difference* (negative) and *substitution* (positive).

6 CONCLUSION

We presented three different algorithms to create variations of one bar drum rhythm patterns as well as the extension of an interface prototype. The aim of the prototype is to support EDM producers and performers to find suitable drum patterns. We used the prototype to test and evaluate the variation algorithms by means of two studies: A series of qualitative expert interviews and a quantitative web-based survey. The expert interviews show that the interaction concept of the prototype is something most participants can imagine working with. It also implies that the acceptance of such a tool in a studio environment would be high, while concerns were raised about precision and reliability when it comes to live performance scenarios. The patterns created by the database-based approach (*DB*) and the neural-network-based method (*RBM*) were mostly considered musical and in many cases perceived to reflect the basic rhythmic idea of the seed pattern. While the genetic algorithm (*GEN*) produced usable patterns in many cases, it was considered more suitable for fills and creative exploration. The web-based study, using an expert-created baseline, allows interesting insights that support the findings of the expert interviews: *GEN* produces patterns suitable for fills that have a tendency to be more different and interesting than the ones produced by *RBM* or *DB*, which in turn are more conservative and suitable as substitute patterns for basic rhythms. The findings of the two studies support each other and shed light on the properties of the compared methods as well as on the perception of rhythm variations of users in general.

Accompanying materials covering the raw survey data, images and audio renderings of the survey patterns,

the UI prototype, a short demo video, and the training configuration for the RBM are available at: <https://github.com/GiantSteps/rhythm-pattern-variation-study>

7 ACKNOWLEDGMENTS

This work is supported by the European Union's seventh Framework Programme FP7 / 2007-2013 for research, technological development, and demonstration under grant agreement no. 610591 (GiantSteps). We thank Stefan Latner for providing help with the Lrn2 framework as well as giving hints and support for training the RBM. We gratefully acknowledge the support of NVIDIA Corporation with the donation of one Titan Black GPU used for this research.

8 REFERENCES

- [1] E. Battenberg and D. Wessel, "Analyzing Drum Patterns Using Conditional Deep Belief Networks," in *Proc. 13th Intl. Society for Music Information Retrieval Conf.* (2012).
- [2] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent, "Modeling Temporal Dependencies in High-Dimensional Sequences: Application to Polyphonic Music Generation and Transcription," in *Proc. 29th Intl. Conf. on Machine Learning* (2012).
- [3] C. Dittmar and M. Pfleiderer, "Automated Estimation of Ride Cymbal Swing Ratios in Jazz Recordings," in *Proc. 16th Intl. Society for Music Information Retrieval Conf.* (2015).
- [4] H. Goh, N. Thome, and M. Cord, "Biasing Restricted Boltzmann Machines to Manipulate Latent Selectivity and Sparsity," in *NIPS Workshop on Deep Learning and Unsupervised Feature Learning* (2010).
- [5] M. Gruhne and C. Dittmar, "Improving Rhythmic Pattern Features Based on Logarithmic Preprocessing," presented at the *126th Convention of the Audio Engineering Society* (2009 May), convention paper 7817.
- [6] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A Fast Learning Algorithm for Deep Belief Nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554 (2006), <http://dx.doi.org/10.1162/neco.2006.18.7.1527>.
- [7] A. Holzapfel and Y. Stylianou, "Scale Transform in Rhythmic Similarity of Music," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 1, pp. 176–185 (2011), <http://dx.doi.org/10.1109/TASL.2010.2045782>.
- [8] J. H. Jensen, M. G. Christensen, and S. H. Jensen, "A Tempo-Insensitive Representation of Rhythmic Patterns," in *Proc. 17th European Signal Processing Conf.* (2009).
- [9] M. A. Kaliakatsos-Papakostas, A. Floros, and M. N. Vrahatis, "evoDrummer: Deriving Rhythmic Patterns through Interactive Genetic Algorithms," in *Evolutionary and Biologically Inspired Music, Sound, Art and Design*, vol. 7834 of *LNCS*, pp. 25–36 (Springer, 2013), http://dx.doi.org/10.1007/978-3-642-36955-1_3.
- [10] C. Lewis, "Using the "Thinking-Aloud" Method in Cognitive Interface Design," Tech. Rep. RC-9265, IBM TJ Watson Research Center (1982).
- [11] U. Marchand and G. Peeters, "Swing Ratio Estimation," in *Proc. 18th Intl. Conf. on Digital Audio Effects* (2015).
- [12] C. Ó Nuanáin, P. Herrera, and S. Jordà, "Target-Based Rhythmic Pattern Generation and Variation with Genetic Algorithms," in *Proc. 12th Sound and Music Computing Conf.* (2015).
- [13] J.-F. Paiement, Y. Grandvalet, S. Bengio, and D. Eck, "A Generative Model for Rhythms," in *NIPS Workshop on Brain, Music and Cognition* (2007).
- [14] D. Shiffman, S. Fry, and Z. Marsh, *The Nature of Code* (Daniel Shiffman, 2012).
- [15] G. Soros and C. Guedes, "Complexity Driven Recombination of MIDI Loops," in *Proc. 12th Intl. Society for Music Information Retrieval Conf.* (2011).
- [16] G. Sioros, A. Holzapfel, and C. Guedes, "On Measuring Syncopation to Drive an Interactive Music System," in *Proc. 13th Intl. Society for Music Information Retrieval Conf.* (2012).
- [17] P. Smolensky, "Information Processing in Dynamical Systems: Foundations of Harmony Theory," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1*, pp. 194–281 (MIT Press, 1986).
- [18] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *J. Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958 (2014).
- [19] T. Tieleman and G. Hinton, "Using Fast Weights to Improve Persistent Contrastive Divergence," in *Proc. 26th Intl. Cons. on Machine Learning* (2009), <http://dx.doi.org/10.1145/1553374.1553506>.
- [20] G. Toussaint, "A Comparison of Rhythmic Similarity Measures," in *Proc. 5th Intl. Cons. on Music Information Retrieval* (2004).
- [21] R. Vogl and P. Knees, "An Intelligent Musical Rhythm Variation Interface," in *Companion Publication 21st Intl. Cons. on Intelligent User Interfaces* (2016), <http://dx.doi.org/10.1145/2876456.2879471>.

THE AUTHORS



Richard Vogl



Matthias Leimeister



Cárthach Ó Nuanáin



Sergi Jordà



Michael Hlatky



Peter Knees

Richard Vogl is a Ph.D. researcher at the Dept. of Computational Perception of the Johannes Kepler University Linz in Austria. His research is focused on rhythm and drum analysis and machine learning.

Matthias Leimeister is a member of the music information research team at Native Instruments, Berlin. His research interests include rhythm analysis, automatic transcription, and music recommendation.

Cárthach Ó Nuanáin is a Ph.D. researcher with the Music and Multimodal Interaction team with the Music Technology Group at Universitat Pompeu Fabra, Barcelona. His research interests include electronic music, algorithmic composition, interface technology, and interaction.

Sergi Jordà holds a B.S. in fundamental physics and a Ph.D. in computer science and digital communication. He

is a senior researcher with the Music Technology Group at Universitat Pompeu Fabra in Barcelona, where he directs the Music and Multimodal Interaction team, and an Associate Professor in the same university.

Michael Hlatky works as a product designer in the areas of interaction design, user interface design and experience design. Before joining Native Instruments, he worked as a sound designer and DSP developer for Audi AG and Bang&Olufsen a/s.

Peter Knees holds a doctorate degree in computer science and is currently assistant professor of the Dept. of Computational Perception of the Johannes Kepler University Linz in Austria. For over a decade, he has been an active member of the music information retrieval research community, branching out to the related areas of multimedia, text IR, recommender systems, and digital media arts.