Considering Vertical and Horizontal Context in Corpus-based Generative Electronic Dance Music

Arne Eigenfeldt
School for the Contemporary Arts
Simon Fraser University
Vancouver, BC
Canada

Philippe Pasquier
School of Interactive Arts and Technology
Simon Fraser University
Surrey, BC
Canada

Abstract

We present GESMI (Generative Electronica Statistical Modeling Instrument) – a computationally creative music generation system that produces Electronic Dance Music through statistical modeling of a corpus. We discuss how the model requires complex interrelationships between simple patterns, relationships that span both time (horizontal) and concurrency (vertical). Specifically, we present how context-specific drum patterns are generated, and how auxiliary percussion parts, basslines, and drum breaks are generated in relation to both generated material and the corpus. Generated audio from the system has been accepted for performance in an EDM festival.

Introduction

Music consists of complex relationships between its constituent elements. For example, a myriad of implicit and explicit rules exist for the construction of successive pitches – the rules of melody (Lerdahl and Jackendoff 1983). Furthermore, as music is time-based, composers must take into account how the music unfolds: how ideas are introduced, developed and later restated. This is the concept of musical form – the structure of music in time. As these relationships are concerned with a single voice, and thus monophonic, we can consider them to be horizontal[1].

Similarly, relationships between multiple voices need to be assessed. As with melody, explicit production rules exist for concurrent relationships – harmony – as well as the relationships between melodic motives: polyphony. We can consider these relationships to be vertical (see Figure 1).

[1] The question of whether melody is considered a horizontal or vertical relationship is relative to how the data is presented: in traditional music notation, it would be horizontal; in sequencer (list) notation, it would be vertical. For the purposes of this paper, will assume traditional musical notation.

Music has had a long history of applying generative methods to composition, due in large part to the explicit rules involved in its production. A standard early reference is the *Musikalsches Würfelspiel* of 1792, often attributed to Mozart, in which pre-composed musical sections were assembled by the user based upon rolls of the dice (Chuang 1995); however, the "Canonic" compositions of the late 15th century are even earlier examples of procedural composition. In these works, a single voice was written out, and singers were instructed to derive their own parts from it by rule: for example, singing the same melody delayed by a set number of pulses, or at inversion (Randel 2003).

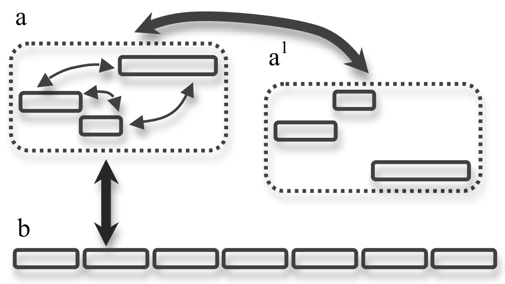


Figure 1. Relationships within three musical phrases, *a, a*^{1}*, b*: melodic (horizontal) between pitches within *a*; formal (horizontal) between *a* and *a*^{1}; polyphonic (vertical) between *a* and *b*.

Exploring generative methods with computers began with some of the first applications of computers in the arts. Hiller's *Illiac Suite* of 1956, created using the Illiac computer at the University of Champaign-Urbana, utilized Markov chains for the generation of melodic sequences (Hiller and Isaacson 1979). In the next forty years, a wide variety of approaches were investigated – see (Papadopoulos and Wiggins 1999) for a good overview of early uses of computers within algorithm composition. However, as the authors suggest, "most of these systems deal with algorithmic composition as a problem solving task rather than a creative and meaningful process". Since that time, this separation has continued: with a few exceptions (Cope 1992, Waschka 2007, Eigenfeldt and Pasquier 2012), contemporary algorithmic systems that employ AI methods remain experimental, rather than generating complete and successful musical compositions.

The same cannot be said about live generative music, sometimes called interactive computer music due to its reliance upon composer or performer input during performance. In these systems (Chadabe 1984, Rowe 1993, Lewis 1999), the emphasis is less upon computational experimentation and more upon musical results. However, many musical decisions – notably formal control and polyphonic relationships – essentially remain in the hands of the composer during performance.

Joel Chadabe was the first to interact with musical automata. In 1971, he designed a complex analog system that allowed him to compose and perform *Ideas of Movement at Bolton Landing* (Chadabe 1984). This was the first instance of what he called interactive composing, "a mutually influential relationship between performer and instrument." In 1977, Chadabe began to perform with a digital synthesizer/small computer system: in *Solo*, the first work he finished using this system, the computer generated up to eight simultaneous melodic constructions, which he guided in realtime. Chadabe suggested that *Solo* implied an intimate jazz group; as such, all voices aligned to a harmonic structure generated by the system (Chadabe 1980).

Although the complexity of interaction increased between the earlier analog and the later digital work, the conception/aesthetic between *Ideas of Movement at Bolton Landing* and *Solo* did not change in any significant way. While later composers of interactive systems increased the complexity of interactions, Chadabe conceptions demonstrate common characteristics of interactive systems:

1. Melodic constructions (horizontal relationships) are not difficult to codify, and can easily be "handed off" to the system;
2. harmonic constructions (vertical relationships) can be easily controlled by aligning voices to a harmonic grid, producing acceptable results;
3. complex relationships between voices (polyphony), as well as larger formal structures of variation and repetition, are left to the composer/performer in realtime.

These limitations are discussed in more detail in Eigenfeldt (2007).

GESMI (Generative Electronica Statistical Modeling Instrument) is an attempt to blend autonomous generative systems with the musical criteria of interactive systems. Informed by methods of AI in generating horizontal relationships (i.e. Markov chains), we apply these methods in order to generate vertical relationships, as well as high-level horizontal relationships (i.e. form) so as to create entire compositions, yet without the human intervention of interactive systems.

The Generative Electronica Research Project (GERP) is an attempt by our research group – a combination of scientists involved in artificial intelligence, cognitive science, machine-learning, as well as creative artists – to generate stylistically valid EDM using human-informed machine-learning. We have employed experts to hand-transcribe 100 tracks in four genres: Breaks, House, Dubstep, and Drum and Bass. Aspects of transcription include musical details (drum patterns, percussion parts, bass lines, melodic parts), timbral descriptions (i.e. "low synth kick, mid acoustic snare, tight noise closed hihat"), signal processing (i.e. the use of delay, reverb, compression and its alteration over time), and descriptions of overall musical form. This information is then compiled in a database, and analysed to produce data for generative purposes. More detailed information on the corpus is provided in (Eigenfeldt and Pasquier 2011).

Applying generative procedures to electronic dance music is not novel; in fact, it seems to be one of the most frequent projects undertaken by nascent generative musician/programmers. EDM's repetitive nature, explicit forms, and clearly delimited style suggest a parameterized approach.

Our goal is both scientific and artistic: can we produce complete musical pieces that are modeled on a corpus, and indistinguishable from that corpus' style? While minimizing human/artistic intervention, can we extract formal procedures from the corpus and use this data to generate all compositional aspects of the music so that a perspicacious listener of the genre will find it acceptable? We have already undertaken empirical validation studies of other styles of generative music (Eigenfeldt et al 2012), and now turn to EDM.

It is, however, the artistic purpose that dominates our motivation around GESMI. As the authors are also composers, we are not merely interested in creating test examples that validate methods. Instead, the goals remain artistic: can we generate EDM tracks and produce a full-evening event that is artistically satisfying, yet entertaining for the participants? We feel that we have been successful, even at the current stage of research, as output from the system has been selected for inclusion in an EDM concert[2] as well as a generative art festival[3].

Related Work

Our research employs several avenues that combine the work of various other researchers. We use Markov models to generate horizontal continuations, albeit with contextual constraints placed upon the queries. These constraints are learned from the corpus, which thus involve machine-learning. Lastly, we use a specific corpus, expert-transcribed EDM in order to generate style-specific music.

Markov models offer a simple and efficient method of deriving correct short sequences based upon a specific corpus (Pachet et al. 2011), since they are essentially quoting portions of the corpus itself. Furthermore, since the models are unaware of any rules themselves, they can be quickly adapted to essentially "change styles" by switching the corpus. However, as Ames points out (Ames 1989), while simple Markov models can reproduce the surface features

2 http://www.metacreation.net/mumewe2013/

3 http://xcoax.org/

of a corpus, they are poor at handling higher-level musical structures. Pachet points out several limitations of Markov-based generation, and notes how composers have used heuristic measures to overcome them (Pachet et al. 2011). Pachet's research aims to allow constraints upon selection, while maintaining the statistical distribution of the initial Markov model. We are less interested in maintaining this distribution, as we attempt to explore more unusual continuations for the sake of variety and surprise.

Using machine-learning for style modeling has been researched previously (Dubnov et al. 2003), however, their goals were more general in that composition was only one of many possible suggested outcomes from their initial work. Their examples utilized various monophonic corpora, ranging from "early Renaissance and baroque music to hard-bop jazz", and their experiments were limited to interpolating between styles rather than creating new, artistically satisfying music. Nick Collins has used music information retrieval (MIR) for style comparison and influence tracking (Collins 2010).

The concept of style extraction for reasons other than artistic creation has been researched more recently by Tom Collins (Collins 2011), who tentatively suggested that, given the state of current research, it may be possible to successfully generate compositions within a style, given an existing database.

Although the use of AI within the creation of EDM has been, so far, mainly limited to drum pattern generation (for example, Kaliakatsos-Papakostas et al. 2013), the use of machine-learning within the field has been explored: see (Diakopoulos 2009) for a good overview. Nick Collins has extensively explored various methods of modeling EDM styles, including 1980s synth-pop, UK Garage, and Jungle (Collins 2001, 2008).

Our research is unique in that we are attempting to generate full EDM compositions using completely autonomous methods informed by AI methods.

Description

We have approached the generation of EDM as a producer of the genres would: from both a top-down (i.e. form and structure) and bottom-up (i.e. drum patterns) at the same time. While a detailed description of our formal generation is not possible here (see Eigenfeldt and Pasquier 2013 for a detailed description of our evolutionary methods for form generation), we can mention that an overall form is evolved based upon the corpus, which determines the number of individual patterns required in all sixteen instrumental parts, as well as their specific relationships in time. It is therefore known how many different patterns are required for each part, and which parts occur simultaneously – and thus require vertical dependencies – and which parts occur consecutively, and thus require horizontal dependencies.

The order of generation is as follows:

1. Form – the score, determining which instruments are active for specific phrases, and their pattern numbers;
2. Drum Patterns – also called *beats*[4] (kick, snare, closed hihat, open hihat);
3. Auxiliary percussion – (ghost kick/snare, cymbals, tambourine, claps, shakers, percussive noises, etc.) generation is based upon the concurrent drum patterns;
4. Bassline(s) – onsets are based upon the concurrent drum pattern, pitches are derived from associated data;
5. Synth and other melodic parts – onsets are based upon bassline, pitches are derived from associated data. All pitch data is then corrected according to an analysis of the implied harmony of the bassline (not discussed here);
6. Drum breaks – when instruments stop (usually immediately prior to a phrase change, and a pattern variation (i.e. drum fill) occurs;
7. One hits – individual notes and/or sounds that offer colour and foreground change that are not part of an instrument's pattern (not discussed here).

Drum Pattern Generation

Three different methods are used to generate drum patterns, including:

1. Zero-order Markov generation of individual subparts (kick, snare, closed hihat, and open hihat);
2. first-order Markov generation of individual subparts;
3. first-order Markov generation of combined subparts.

In the first case, probabilities for onsets on a given beat subdivision (i.e. sixteen subdivisions per four beat measure) are calculated for each subpart based upon the selected corpus (see Figure 2). As with all data derived from the corpus, the specific context is retained. Thus, if a new drum pattern is required, and it first appears in the main verse (section C), only data derived from that section is used in the generation.

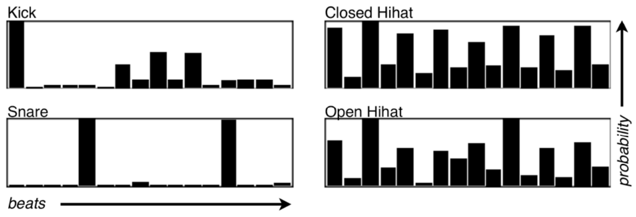


Figure 2. Onset probabilities for individual subparts, one measure (sixteenth-note subdivisions), main verse (C section), "Breaks" corpus.

In the second case, data is stored as subdivisions of the quarter note, as simple on/off flags (i.e. 1 0 1 0) for each subpart, and separate subparts are calculated independ-

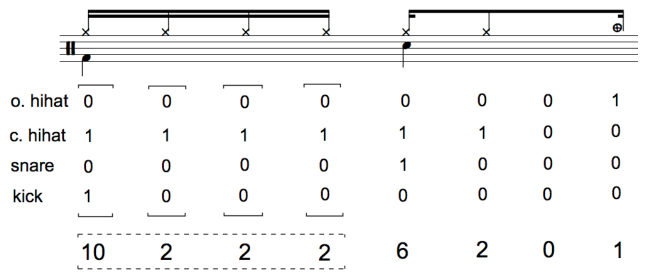
[4] The term "beat" has two distinct meanings. In traditional music, beat refers to the basic unit of time – the pulse of the music – and thus the number of subdivisions in a measure; within EDM, beat also refers to the combined rhythmic patterns created by the individual subparts of the drums (kick drum, snare drum, hi-hat), as well as any percussion patterns.

ently. Continuations[5] are considered across eight measure phrases, rather than limited to specific patterns: for example, the contents of an eight measure pattern are considered as thirty-two individual continuations, while the contents of a one measure pattern that repeats eight times are considered as four individual continuations with eight instances, because they are heard eight separate times. As such, the inherent repetition contained within the music is captured in the Markov table.

In the third case, data is stored as in the second method just described; however, each subpart is considered 1 bit in a 4-bit nibble for each subdivision that encodes the four subparts together:

bit 1 = open hihat;
bit 2 = closed hihat;
bit 3 = snare;
bit 4 = kick.

This method ensures that polyphonic relationships between parts – vertical relationships – are encoded, as well as time-based relationships – horizontal relationships (see Figure 3).



o. hihat	0	0	0	0	0	0	0	1
c. hihat	1	1	1	1	1	1	0	0
snare	0	0	0	0	1	0	0	0
kick	1	0	0	0	0	0	0	0
	10	2	2	2	6	2	0	1

Figure 3. Representing the 4 drum subparts (of two beats), as a 4-bit nibble (each column of the four upper rows), translated to decimal (lower row), for each sixteenth-note subdivision. These values are stored as 4-item vectors representing a single beat.

It should be noted that EDM rarely, if ever, ventures outside of sixteenth-note subdivisions, and this representation is appropriate for our entire corpus.

The four vectors are stored, and later accessed, contextually: separate Markov tables are kept for each of the four beats of a measure, and for separate sections. Thus, all vectors that occur on the second beat are considered queries to continuations for the onsets that occur on the third beat; similarly, these same vectors are continuations for onsets that occur on the first beat. The continuations are stored over eight measure phrases, so the first beat of the second measure is a continuation for the fourth beat of the first measure. We have not found it necessary to move beyond first-order Markov generation, since our data involves four-items representing four onsets.

We found that the third method produced the most accurate re-creations of drum patterns found in the corpus, yet the first method produced the most surprising, while maintaining usability. Rather than selecting only a single method for drum pattern generation, it was decided that the three separate methods provided distinct "flavors", allowing users several degrees of separation from the original corpus. Therefore, all three methods were used in the generation of a large (>2000) database of potential patterns, from which actual patterns are contextually selected. See (Eigenfeldt and Pasquier 2013) for a complete description of our use of populations and the selection of patterns from these populations.

[5] The generative music community uses the term "continuations" to refer to what is usually called transitions (weighted edges in the graph).

Auxiliary Percussion Generation

Auxiliary percussion consists of non-pitched rhythmic material not contained within the drum pattern. Within our corpus, we have extracted two separate auxiliary percussion parts, each with up to four subparts. The relationship between these parts to the drum pattern is intrinsic to the rhythmic drive of the music; however, there is no clear or consistent musical relationship between these parts, and thus no heuristic method available for their generation.

We have chosen to generate these parts through first-order Markov chains, using the same contextual beat-specific encoding just described; as such, logical horizontal relationships found in the corpus are maintained. Using the same 4-bit representation for each auxiliary percussion part as described in method 3 for drum pattern generation, vertical consistency is also imparted; however, the original relationship to the drum pattern is lost. Therefore, we constrain the available continuations.

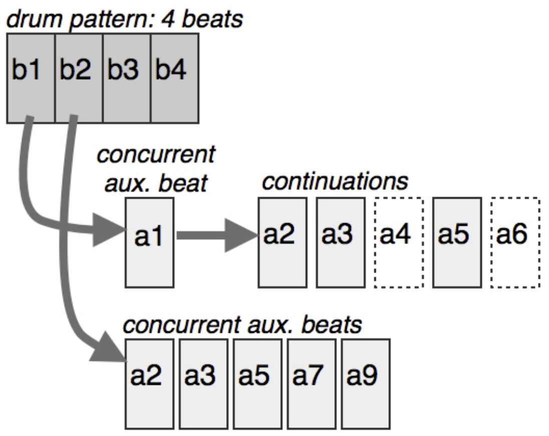


Figure 4. Maintaining contextual vertical and horizontal relationships between auxiliary percussion beats (*a*) and drum beats (*b*).

As the drum patterns are generated prior to the auxiliary percussion, the individual beats from these drum patterns serve as the query to a cross-referenced transition table made up of auxiliary percussion pattern beats (see Figure 4). Given a one measure drum pattern consisting of four beats *b1 b2 b3 b4*, all auxiliary percussion beats that occur simultaneously with *b1* in the corpus are considered as available concurrent beats for the auxiliary percussion pattern's initial beat. One of these, *a1*, is selected as the first beat, using a weighted probability selection. The available continuations for *a1* are *a2-a6*. Because the next auxiliary percussion beat must occur at the same time as the drum pattern's *b2*, the auxiliary percussion beats that occur concurrently with *b2* are retrieved: *a2, a3, a5, a7, a9*. Of these, only *a2, a3,* and *a5* intersect both sets; as such, the available continuations for *a1* are constrained, and the next auxiliary percussion beat is selected from *a2, a3,* and *a5*.

Of note is the fact that any selection from the constrained set will be horizontally correct due to the transition table, as well as being vertically consistent in its relationship to the drum pattern due to the constraints; however, since the selection is made randomly from the probabilistic distribution of continuations, the final generated auxiliary percussion pattern will not necessarily be a pattern found in the corpus.

Lastly, we have not experienced insufficient continuations since we are working with individual beats, rather than entire measures: while there are only a limited number of four-element combinations that can serve as queries, a high number of 1-beat continuations exist.

Bassline Generation

Human analysis determined there were up to two different basslines in the analysed tracks, not including bass drones, which are considered a synthesizer part. Bassline generation is a two-step process: determining onsets (which include held notes longer than the smallest quantized value of a sixteenth note); then overlaying pitches onto these onsets.

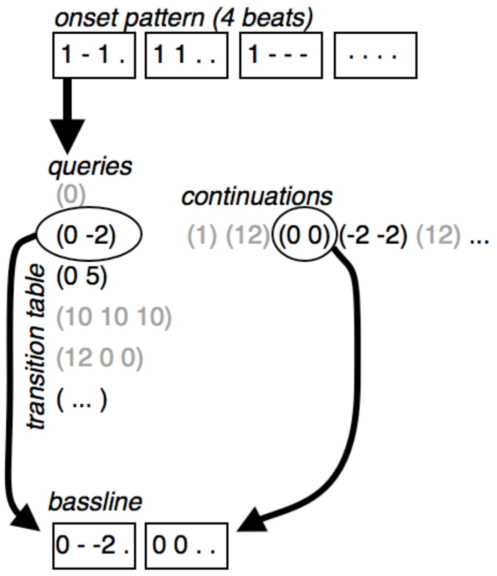


Figure 5. Overlaying pitch-classes onto onsets, with continuations constrained by the number of pitches required in the beat.

Bassline onset generation uses the same method as that of auxiliary percussion – contextually dependent Markov sequences, using the existing drum patterns as references. One Markov transition table encoded from the corpus' basslines contains rhythmic information: onsets (1), rests (.), and held notes (-). The second transition table contains only pitch data: pitch-classes relative to the track's key (-24 to +24). Like the auxiliary percussion transition tables, both the queries and the continuations are limited to a single beat.

Once a bassline onset pattern is generated, it is broken down beat by beat, with the number of onsets occurring within a given beat serving as the first constraint on pitch selection (see Figure 5). Our analysis derived 68 possible 1-beat pitch combinations within the "Breaks" corpus. In Figure 5, an initial beat contains 2 onsets (`1 – 1 .`) Within the transition table, 38 queries contain two values (not grayed out in Figure 5's vertical column): one of these is selected as the pitches for the first beat using a weighted probability selection (circled). As the next beat contains 2 onsets (`1 1 . .`), the first beat's pitches (`0 -2`) serve as the query to the transition table, and the returned continuations are constrained by matching the number of pitches required (not grayed out in Figure 5's horizontal row). One of these is selected for the second beat (circled) using additional constraints described in the next section. This process continues, with pitch-classes being substituted for onset flags (bottom).

Additional Bassline Constraints

Additional constraints are placed upon the bassline generation, based upon user set "targets". These include constraints the following:

– Density: favouring fewer or greater onsets per beat;
– straightness: favouring onsets on the beat versus syncopated;
– dryness: favouring held notes versus rests;
– jaggedness: favouring greater or lesser differentiation between consecutive pitch-classes.

Each available continuation is rated in comparison to the user-set targets using a Euclidean distance function, and an exponential random selection is made from the top 20% of these ranked continuations.

This notion of targets appears throughout the system. While such a method does allow some control over the generation, the main benefit will be demonstrated in the next stage of our research: successive generations of entire compositions – generating hour long sets of tracks, for example – can be guaranteed to be divergent by ensuring targets for parameters are different between runs.

Contextual Drum-fills

Fills, also known as drum-fills, drum-breaks, or simply breaks, occur at the end of eight measure phrases as variations of the overall repetitive pattern, and serve to highlight the end of the phrase, and the upcoming section change. Found in most popular music, they are often restricted to the drums, but can involve other instruments (such as auxiliary percussion), as well as a break, or silence, from the other parts.

Fills are an intuitive aspect of composition in pattern-based music, and can be conceptually reduced to a rhythmic variation. As such, they are not difficult to code algorithmically: for example, following seven repetitions of a one measure drum pattern, a random shuffle of the pattern will produce a perfectly acceptable fill for the eighth measure (see Figure 6).

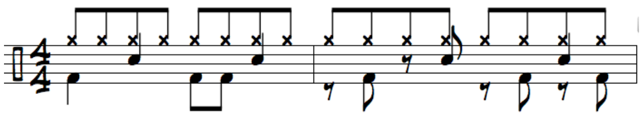


Figure 6. Left: drum pattern for kick, snare, and hihat; right: pattern variation by shuffling onsets can serve as a fill.

Rather than utilizing such creative "shortcuts", our fill generation is based entirely upon the corpus. First, the *location* of the fill is statistically generated based upon the location of fills within phrases in the corpus, and the generated phrase structure. Secondly, the *type* of fill is statistically generated based upon the analysis: for example, the described pattern variation using a simple onset shuffle has a 0.48 probability of occurring within the Breaks corpus – easily the most common fill type. Lastly, the actual variation is based upon the specific *context*.

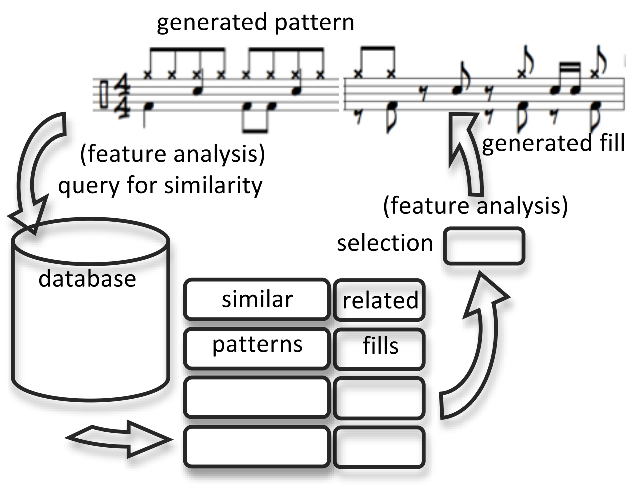


Figure 7. Fill generation, based upon contextual similarity

Fills always replace an existing pattern; however, the actual pattern to be replaced within the generated drum part may not be present in the corpus, and thus no direct link would be evident from a fill corpus. As such, the original pattern is analysed for various features, including *density* (the number of onsets) and *syncopation* (the percentile of onsets that are not on strong beats). These values are then used to search the corpus for patterns with similar features. One pattern is selected from those that most closely match the query. The relationship between the database's pattern and its fill is then analysed for *consistency* (how many onsets remain constant), *density change* (how many onsets are added or removed), and *syncopation change* (the percentile change in the number of onsets that are not on strong beats). This data is then used to generate a variation on the initial pattern (see Figure 7).

The resulting fill will display a relationship to its original pattern in a contextually similar relationship to the corpus.

Conclusions and Future Work

The musical success of EDM lies in the interrelationship of its parts, rather than the complexity of any individual part. In order to successfully generate a complete musical work that is representative of the model, rather than generating only components of the model (i.e. a single drum pattern), we have taken into account both horizontal relationships between elements in our use of a Markov model, as well as vertical relationships in our use of constraint-based algorithms. Three different methods to model these horizontal and vertical dependencies at generation time have been proposed in regards to drum pattern generation (through the use of a combined representation of kick, snare, open and closed hihat, as well as context-dependent Markov selection), auxiliary percussion generation (through the use of constrained Markov transitions) and bassline generation (through the use of both onset- and pitch-constrained Markov transitions. Each of these decisions contributes to what we believe to be a more successful generation of a complete work that is stylistically representative and consistent.

Future work includes validation to investigate our research objectively. We have submitted our work to EDM festivals and events that specialize in algorithmic dance music, and our generated tracks have been selected for presentation at two festivals so far. We also plan to produce our own dance event, in which generated EDM will be presented alongside the original corpus, and use various methods of polling the audience to determine the success of the music.

Lastly, we plan to continue research in areas not discussed in this paper, specifically autonomous timbral selection and signal processing, both of which are integral to the success of EDM.

This research was created in MaxMSP and Max4Live running in Ableton Live. Example generations can be heard at soundcloud.com/loadbang.

Acknowledgements

This research was funded by a grant from the Canada Council for the Arts, and the Natural Sciences and Engineering Research Council of Canada.

References

Ames, C. 1989. *The Markov Process as a Compositional Model: A Survey and Tutorial.* Leonardo 22(2).

Chadabe, J. 1980. *Solo: A Specific Example of Realtime Performance.* Computer Music - Report on an International Project. Canadian Commission for UNESCO.

Chadabe, J. 1984. *Interactive Composing.* Computer Music Journal 8:1.

Chuang, J. 1995. Mozart's Musikalisches Würfelspiel, http://sunsite.univie.ac.at/Mozart/dice/, retrieved September 10, 2012.

Collins, N. 2001. *Algorithmic composition methods for breakbeat science*. Proceedings of Music Without Walls, De Montfort University, Leicester, 21-23.

Collins, N. 2008. *Infno: Generating synth pop and electronic dance music on demand.* Proceedings of the International Computer Music Conference, Belfast.

Collins, N. 2010. *Computational analysis of musical influence: A musicological case study using MIR tools.* Proceedings of the International Symposium on Music Information Retrieval, Utrecht.

Collins, T. 2011. *Improved methods for pattern discovery in music, with applications in automated stylistic composition.* PhD thesis, Faculty of Mathematics, Computing and Technology, The Open University.

Cope, D. 1992. *Computer Modeling of Musical Intelligence in EMI.* Computer Music Journal, 16:2, 69–83.

Diakopoulos, D., Vallis, O., Hochenbaum, J., Murphy, J., and Kapur, A. 2009. 21st Century Electronica: MIR Techniques for Classification and Performance. In: Proceedings of the International Society for Music Information Retrieval Conference (ISMIR), Kobe, 465–469.

Dubnov, S., Assayag, G., Lartillot, O. and Bejerano, G. 2003. *Using machine-learning methods for musical style modeling*. Computer, 36:10.

Eigenfeldt, A. 2007. *Computer Improvisation or Real-time Composition: A Composer's Search for Intelligent Tools.* Electroacoustic Music Studies Conference 2007, http://www.ems-network.org/spip.php?article264 Accessed 3 February 2013.

Eigenfeldt, A. 2012 *Embracing the Bias of the Machine: Exploring Non-Human Fitness Functions*. Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, Palo Alto.

Eigenfeldt, A. 2013. *Is Machine Learning the Next Step in Generative Music?* Leonardo Electronic Almanac, Special Issue on Generative Art, forthcoming.

Eigenfeldt, A., and Pasquier, P. 2011. *Towards a Generative Electronica: Human-Informed Machine Transcription and Analysis in MaxMSP*. Proceedings of the Sound and Music Computing Conference, Padua.

Eigenfeldt, A., and Pasquier, P. 2012. *Populations of Populations - Composing with Multiple Evolutionary Algorithms*. P. Machado, J. Romero, and A. Carballal (Eds.). In: EvoMUSART 2012, LNCS 7247, 72–83. Springer, Heidelberg.

Eigenfeldt, A., Pasquier, P., and Burnett, A. 2012. *Evaluating Musical Metacreation.* International Conference of Computational Creativity, Dublin, 140–144.

Eigenfeldt, A., and Pasquier, P. 2013. *Evolving Structures in Electronic Dance Music,* GECCO 2013, Amsterdam.

Hiller, L., and Isaacson, L. 1979. *Experimental Music; Composition with an Electronic Computer.* Greenwood Publishing Group Inc. Westport, CT, USA.

Kaliakatsos-Papakostas, M., Floros, A., and Vrahatis, M.N. 2013. EvoDrummer: Deriving rhythmic patterns through interactive genetic algorithms. In: Evolutionary and Biologically Inspired Music, Sound, Art and Design. Lecture Notes in Computer Science Volume 7834, 2013, pp 25–36.

Lerdahl, F., Jackendoff, R. 1983. *A generative theory of tonal music.* The MIT Press.

Lewis, G. 1999. *Interacting with latter-day musical automata.* Contemporary Music Review, 18:3.

Pachet, F., Roy, P., and Barbieri, G. 2011. *Finite-length Markov processes with constraints.* Proceedings of the Twenty-Second international joint conference on Artificial Intelligence Volume One. AAAI Press.

Papadopoulos, G., and Wiggins, G. 1999. *AI methods for algorithmic composition: A survey, a critical view and future prospects.* In: AISB Symposium on Musical Creativity, 110–117, Edinburgh, UK.

Randel, D. 2003. *The Harvard Dictionary of Music.* Belknap Press.

Rowe, R. 1993. *Interactive Music Systems*. Cambridge, Mass., MIT Press.

Waschka, R. 2007. *Composing with Genetic Algorithms: GenDash.* Evolutionary Computer Music, Springer, London, 117–136.