Evolving Structures for Electronic Dance Music

Arne Eigenfeldt
School for the Contemporary Arts
Simon Fraser University
Vancouver, Canada
arne_e@sfu.ca

Philippe Pasquier
School of Interactive Arts and Technology
Simon Fraser University
Surrey, Canada
pasquier@sfu.ca

ABSTRACT
We present GESMI (Generative Electronica Statistical Modeling Instrument), a software system that generates Electronic Dance Music (EDM) using evolutionary methods. While using machine learning, GESMI rests on a corpus analysed and transcribed by domain experts. We describe a method for generating the overall form of a piece and individual parts, including specific patterns sequences, using evolutionary algorithms. Lastly, we describe how the user can use contextually-relevant target features to query the generated database of strong individual patterns. As our main focus is upon artistic results, our methods themselves use an iterative, somewhat evolutionary, design process based upon our reaction to results.

Categories and Subject Descriptors
H.5.5 [**Sound and Music Computing]:** Methodologies and techniques.

General Terms
Design, Human Factors.

Keywords
Generative Music, Evolutionary Art, Electronic Dance Music.

1. INTRODUCTION AND MOTIVATIONS
Computational creativity, or metacreation [22] is the idea of endowing machines with creative behavior. Metacreation, as the contemporary approach to generative art, involves using tools and techniques from artificial intelligence, artificial life, and machine learning (themselves inspired by cognitive and life sciences) to develop software systems that are creative on their own.

Musical metacreation can be broken down into a number of canonical problems:

1. **Composition** – being the process of creating a series of performance instructions for musical performers, (i.e. a score);
2. **Interpretation** – being the process of performing a musical composition and producing an audio rendering;
3. **Improvisation** – which combines (1) and (2) in real-time performance;
4. **Accompaniment** – being the process of following a live performer in an accompanying role, possibly performing pre-composed music;
5. **Continuation** – being the process of continuing a given musical input in the same style.

GECCO'13, July 6–10, 2013, Amsterdam, The Netherlands.

Metacreative systems can be corpus-based or being generating from scratch, and inputs/outputs can be either raw audio signal or symbolic notations (i.e. a musical score, MIDI file, a text file). While human-competitive results have long been achieved for corpus-based composition [5], interpretation is a significantly harder problem. While artificial intelligence has been tremendously successful at emulating cognitive operations (like composition), human-level physical actions (such as playing an instrument) is still out of reach. This is demonstrated by the RENCON workshops[1], in which computer systems attempt to generate expressive musical performance from musical scores, with only moderate success.

The vast majority of the literature on computational creativity is looking at composition and often reaches a human-competitive level for short excerpts. However, long-term dependencies and overall structure are harder to capture. In this work, we tackle the problem of generating complete pieces of EDM music using an audio corpus that has been transformed into a symbolic corpus by human experts. EDM music is an appropriate domain to tackle for the purposes of musical metacreation, as the interpretation problem is mitigated: EDM practitioners typically use machines (software and hardware) to directly produce audio signals. This simplifies the interpretation problem quite significantly.

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 beats, percussion parts, bass lines, melodic parts), timbral descriptions of all parts (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.

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. As with many cases of creative modeling, initial success will tend to be encouraging to the artist: generating beats, bass lines, and synth parts that *resemble* specific dance genres is not that difficult. However, progressing to a stage where complete pieces are generated that are indiscernible from the model is another matter. In those cases, the "artistic voice" argument tends to emerge: why spend the enormous effort required to accurately emulate someone else's music, when one can easily insert algorithms that reflect one's personal aesthetic? It is indeed possible to generate music that is, in such cases, merely *influenced* by the corpus. While this may prove more artistically

[1] http://renconmusic.org/

satisfying than emulation, we suggest that both are needed. The proposed system, GESMI (Generative Electronica Statistical Modeling Instrument), through user formulated high level queries, allows for the close emulation of the corpus as well as the exploration of its surroundings, thus allowing for potential unexpected directions in composition.

The work described in this paper is part of the effort of a research group[2] that is actively investigating computational creativity. 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 [9], 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?

In the following, we describe the use of evolutionary algorithms (EA) to generate the overall form of complete pieces (Sections 3 and 3.1), specifically the generation of form-states, which are the states of specific parts within that form (Section 3.2), the determination of how form-states become specific patterns (Section 3.2.3), and a description of how patterns, both formal and part-specific, are selected from a population of pre-evolved forms and patterns (Section 4). Figure 1 presents the overall structure of the program. GESMI starts by generating an overall form for the piece, and then determines form-states for each part, after which it generates a database of valid patterns and assembles these based on the user preferences. As such, GESMI has a hybrid architecture that behave in both top-down and bottom-up fashion.

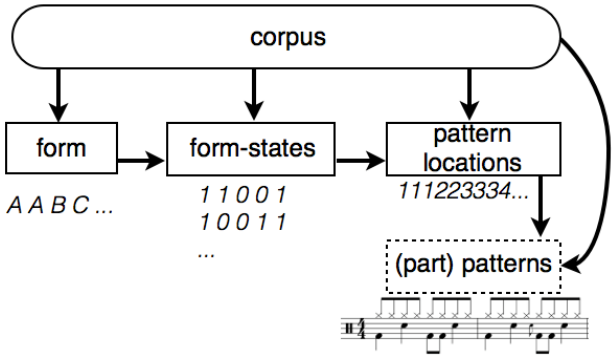

Figure 1. Program structure. The corpus is directly used in the generation of form, form-states, pattern-locations, and individual part patterns (not described here).

2. BACKGROUND AND RELATED WORK

Evolutionary computation has been used within music for over two decades in various ways. Todd and Werner [19] provide a good overview of the earlier musical explorations using such approaches, while Miranda and Biles [17] provide a more recent survey. Very few of these approaches have been compositional in nature; instead, their foci have tended to be studies, while ours are the generation of complete musical compositions.

Several real-time applications of GAs have been used, including [21], which selected individuals from an Interactive Genetic Algorithm (IGA) suitable for the immediate situation within a real-time improvisation. Another approach was by Beyls [1] in which the fitness function sought either similar individuals or contrasting individuals to an immediate situation within an improvisation.

Thywissen [18] describes a system that allows composers to evolve musical structures interactively. Of note is the consideration of higher-level musical structures (i.e. Form), which he calls meta-compositional grammars.

Waschka [20] used a GA to generate contemporary art music. His explanation of the relationship of time within music is fundamental to understanding the potential for evolutionary algorithms within art-music: "unlike material objects, including some works of art, music is time-based. The changes heard in a piece over its duration and how those changes are handled can be the most important aspect of a work." Waschka's *GenDash* has several important attributes, a number of which are unusual: an individual is a measure of music; all individuals in all generations are performed; the fitness function is random, leading to random selection; the composer chooses the initial population. Of note is the second stated attribute, the result of which is that "the evolutionary process itself, not the result of a particular number of iterations, constituted the music". Waschka provides some justifications for his heuristic choices, suggesting that while they may not be observed in real-world compositional process, they do provide musically useful results.

Eigenfeldt and Pasquier [7] describe a system in which a population of rhythms is evolved, and the successive generations serve as the unfolding of musical motives. A second population of forms is evolved, which determines how separate rhythmic motives are interconnected.

In summary, EAs have been used successfully in experimental music [1, 8, 10, 16 20] and improvisation [2, 3, 21] for several years. In most cases, researchers and artists have been able to overcome the main difficulty in applying such techniques to music – namely the difficulty of formulating an effective aesthetic fitness function – through a variety of heuristic methods. One particularly attractive feature of EAs to composers relates to the notion of musical development – the evolution of musical ideas over time – and its relationship to biological evolution. As music is a time-based art, the presentation of successive generations – rather than only the final generation – allows for the aural exposition of evolving musical ideas.

However, Electronic Dance Music (EDM), particularly its non-experimental genres, does not offer the same leisurely exposition of multiple ideas as does contemporary concert music or jazz: there are fewer motives, and their presentation requires explicit control [4]. Although more heuristic methods are required in order to successfully generate EDM algorithmically, evolutionary algorithms still offer creative possibilities within the genre.

In terms of EDM, Wooller and Brown describe a system for evolutionary morphing of EDM [23], in which transformation of stylistic material is user-defined through a system of weightings. Evolectronica, an online site, was active until 2010, and was a self-described effort at a "communal music making experience" in

[2] Generative Electronica Research Project: www.metacreation.net

which a community of users rated evolved individuals that consisted of audio loops, in an effort to create "eclectic electro house beats, laid-back deep house, future drum and bass, and brass/wind/strings/percussion". This work eventually grew into DarwinTunes, described more fully in [15].

Although the use of EAs within the creation of EDM has been, so far, somewhat limited, the use of machine-learning within the field has been explored: see [6] for a good overview.

Genetic Algorithms (GAs) have proven to be useful in situations where solutions may be known, but determining specific algorithms in order to achieve those solutions are difficult to conceive. This is the case in artistic situations, in which rules for aesthetic evaluation are often extremely difficult to formulate. For example, one of the specific requirements of our research is to generate individual states for instruments – whether an instrumental part is present during a phrase – within the overall formal structure of an EDM track. The complex relationships between the states of instrumental parts in EDM are intuitively understood as being "correct" by composers; however, this "correctness" cannot be easily translated into an algorithm. An EA proved to be particularly useful in providing a solution (see Section 3).

An intriguing possibility for the application of an EA within music is the exploration of the building block hypothesis [12]: instead of attempting to directly generate a final complex solution – in our case a complete composition – it is possible to generate lower order, yet highly fit, schemata, and then using these schemata to assemble the larger composition. This approach already has been used to successfully generate music for string quartet in which the initial population was derived from a human-composed corpus [10], and initialized individuals were already considered strong. Similarly, selection from a generated population of individual movements was used to assemble a larger multi-movement composition [7]. In the research presented here, we generate a large population of individuals for various musical requirements – forms and states; beat patterns, bass lines – and then select from these populations based upon the contextual requirements. Somewhat counter to most uses of evolutionary algorithms in music, in which structure may evolve organically, we use the products of EAs to assemble structures, which themselves have been generated using tightly controlled evolutionary algorithms.

3. FORM GENERATION

Although EAs have been used to generate beat patterns for EDM [13, 14], we believe that such goals can be accomplished through other methods more effectively. Within EDM, there is a great deal of constraint on what constitutes a successful beat – one that "grooves" [4]. Certain musical expectations must be met – for example, a snare on beats two and four – along with a variable amount of fluidity in all parts. Since the number of possible solutions is actually limited, many other, less complicated, techniques can be used.

We have chosen instead to use probabilistic methods, including Markov chains, to generate actual beat patterns (see Section 4). The results have proven to be extremely successful – replicating the consistency required by EDM, yet offering the necessary originality to maintain interest.

Form, or the unfolding of music in time, provides an opportunity to use EAs. The four EDM styles analysed use 8-bar phrases – comprised mainly of 2-bar patterns – to a very high degree of consistency: for example, less than 2% of the 621 phrases in the Breaks corpus are something other than 8 bars in length. Similarly, there is not a great deal of variation within the individual patterns: repetition, both at the macro- (form) and meso- (pattern) levels, is explicit (see Figure 2). For example, no part (i.e. instrument) in the Breaks corpus contained more than five unique patterns. The complexity of the music is at the micro-level (subtle delays and accents of onsets) from which we appreciate the "groove" [4], as well as how the individual patterns interact, and how these patterns are successively presented. This paper addresses the latter aspects: elements of "groove" are yet to be incorporated.

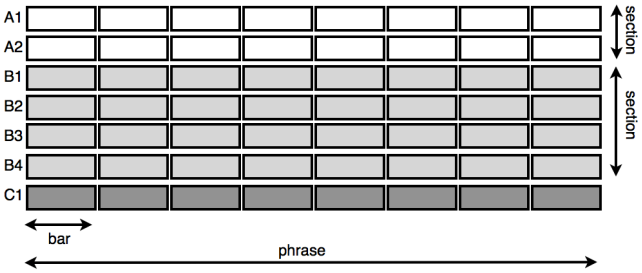

Figure 2. Formal elements within EDM: patterns (1, 2, 4, or 8-bars), phrases (8-bar collections of patterns), and sections (of a variable number of phrases). Shading indicates pattern changes between phrases.

All of the tracks in all four genres within the corpus can be segmented into five distinct sections, which we label A-E:

A- *Lead-in*: the initial section with often only a few parts present;

B- *Intro*: a bridge between the Lead-in and the Verse. More instruments are present than the Lead-in, but not as full as the Verse;

C- *Verse*: the main section of the track, in which all instruments are present, which can occur several times;

D- *Breakdown*: a contrasting section to the verse in which the beat may drop out, or a filter may remove all mid– and high–frequencies. It will tend to build tension, and lead back to the verse;

E- *Outro*: the fade-out of the track.

Furthermore, after close-listening to the tracks within the corpus, the following generalities were determined (for the specific tracks in their specific styles):

– Every track begins with a Lead-in (A), and ends with an Outro (E);

– Sixteen different instrumental parts were identified, including a main drum beat (MBeat), two additional auxiliary percussion parts (Aux1 and Aux2), two different bass parts (Bass and Bass2), two rhythmic synth parts, (RSyn1 and RSyn2), two melodic synth parts (MSyn1 and MSyn2), pads, keys, drone, atmospheric synth, and ancillary melodic parts (Axl, Axl2, Axl3) such as vocals and guitars. These are the total number of unique parts: no track exhibited all sixteen parts within it;

– Parts enter or drop out at the *start* of any phrase. Phrases are constant in terms of their contents: parts rarely drop out or enter *within* a phrase: the exception being drum- and percussion-fills, and one-hits (single events that occur only once);

– Each individual part may contain between 1 and 5 separate patterns over a single track;

– Variety is introduced by one-hits and fills (not discussed here).

	Section	MBeat	Aux1	Aux2	Bass	Bass2	RSyn1	RSyn2	MSyn1	MSyn2	Pads	Keys	Drone	Atmos	Axl	Axl2	Axl3
0	A	0	0	0	1	0	0	0	0	0	0	1	0	0	1	0	0
1	A	0	0	0	1	0	0	0	1	0	1	1	0	0	1	1	0
2	B	1	0	1	1	0	0	0	1	0	0	1	0	0	2	2	0
3	B	1	0	1	1	0	0	0	1	0	0	1	0	0	0	3	0
4	B	1	0	1	1	0	1	0	1	0	0	1	0	0	1	4	0
5	B	1	0	1	1	0	1	0	1	0	0	1	0	0	3	5	1
6	D	1	0	1	1	0	1	0	1	0	0	1	0	0	3	5	1
7	C	1	1	1	1	0	1	0	1	0	0	1	0	0	0	0	2
8	C	1	1	1	1	0	1	0	1	0	0	1	0	0	0	0	3
9	C	1	1	1	1	0	1	0	1	0	1	1	0	0	0	1	4
10	C	1	1	1	1	0	1	0	1	0	1	1	0	0	0	1	4
11	D	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
12	D	1	0	0	0	0	0	0	1	1	0	1	1	0	0	0	0
13	C	1	1	0	1	0	0	0	1	2	0	1	1	0	0	0	0
14	C	1	1	1	1	0	1	0	1	2	1	1	1	0	0	1	0
15	D	1	1	1	0	0	0	0	1	0	1	1	1	0	0	0	0
16	D	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3
17	D	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	4
18	C	1	1	1	1	0	1	0	1	0	0	1	0	0	1	0	5
19	E	1	1	1	1	0	1	0	1	0	1	1	0	0	0	1	4
20	E	1	1	1	1	0	1	0	1	0	1	1	0	0	0	2	3

Figure 3. Complete Form-state (including pattern #) for *Burma*, by Lostep, Sasha Remix radio edit (Breaks corpus). Individual parts are listed above, phrases numbers indicated in the first column, and sections are indicated by letter names in the second column.

In Figure 3, each row represents an 8-bar phrase; each column an instrumental part. Values in columns indicate pattern numbers (stored elsewhere). Zeros indicate the part being tacet (off). This representation thus contains the overall form – in this case, the first ten phrases are AABBBBDCCCC – the individual part's patterns, and their location within the form. We refer to this representation, as shown in Figure 3, as a *form-state*, and the goal of our generative formal structure.

3.1 Generating Form via Markov Model

Form-states seem, at first glance, straightforward and intuitive. However, after some cursory analysis, it becomes quickly apparent that a successful form-state is based upon complex interactions between individual parts, and within the part itself. No heuristic solutions for algorithms presented themselves; however, as rules *could* be determined for successful individuals, we turned to an evolutionary algorithm.

A genetic algorithm was created to generate a population of forms-states modeled on the corpus (see Section 3.2). The first step involves generating an overall form using a first-order Markov model operating in reverse: each section's predecessor, rather than successor, was stored (see Table 1).

Table 1. Markov transition table of a section's predecessor, "Breaks" corpus. Sections at left are preceded by sections above with indicated probability

	A	B	C	D	E
A	1.0				
B	0.35	0.65			
C		0.04	0.76	0.2	
D		0.11	0.31	0.57	0.01
E			0.26	0.02	0.72

The user requests the total number of phrases (the mean number of phrases per piece for the "Breaks" corpus is 27), and the number of sections (the mean number of separate sections for the "Breaks" corpus is 7). A population of 100 individuals is generated, whose genotype consists of the five form values, A-E. Each individual is generated in reverse, beginning with E, using the Markov table for probabilistic continuations. When the generation arrives at A (the target), a test is run to determine if the length of the last section (A) is within a range of acceptability: a heuristically chosen value (0.05) around the corpus' mean (0.102 of the total length of the track). If the individual is within the acceptable range, the individual is reversed, and generation is complete; if not, generation continues. When 100 valid individuals are generated, they are individually rated using a distance function in relation to the user-specified values: the closest matched individual is then selected. As this single generation produces a musically valid and successful form, there is no need to further evolve the population.

3.2 Generating Part States via Genetic Algorithm

Given a generated form (see Section 3.1), states (on or off) for each part can be generated now – the form-state. Initially, we are only generating on or off states, as the actual pattern numbers are generated later (see Section 3.2.3).

3.2.1 Initializing with Strong Individuals

Originally, a population of form-states was randomly generated for the 16 instrumental parts, with initial states being either off (0) or on (1); however, we now use more intelligent methods in order to initialize the population with stronger individuals. Using probabilities derived from the corpus, individual part states are generated that reflect the corpus for that specific part, in terms of probability of appearance in a specific section, likelihood entering in the first phrase of a section, etc. Given the representation of data shown in Figure 3, we consider the each part to have strong *vertical* relationships: consistent within themselves (as each part is displayed in a vertical column); however, *horizontal* relationships – that is, relationships *between* parts, and what we consider to be an important defining aspect of successful EDM – have not been considered. These relationships are evolved through the genetic algorithm.

3.2.2 A Multi-Objective Fitness Function

A multi-objective fitness function evaluates each individual based upon the following criteria, comparing it to data derived from the corpus itself. "Density" refers to whether a part is on for a given phrase, relative to the other parts in a phrase, or to successive phrases within its own part:

– F_1: part density per section (i.e. the ratio of on to off states for parts within a specific section);

– F_2: density changes at section divisions (i.e. how many parts enter or exit at a section change, relative to the last phrase of the previous section);

– F_3: part changes at section divisions (i.e. which parts enter or exit at specific section beginnings);

– F_4-F_{13}: specific individual part relationships (i.e. comparing the overall density of two parts within a section, as well as the percentile of both parts being active at the same time). The part relationships include MBeat-Aux1, Aux1-Aux2, Bass-Bass2, RSyn1-RSyn2, MSyn1-MSyn2, Pads-Keys, Pads-Drone, Drone-Atmos, Axl-Axl2, Axl2-Axl3.

Evaluations are made in comparison to coefficients derived from the analysis of the corpus. For example, a three-phrase section in which a part is off, on, and on (0 1 1) results in the coefficient 0.67, for that part, in that section. All coefficients across all tracks for that section and part are collected from the corpus, and the mean and standard deviation calculated. If the generated part's coefficient for the section is outside the range of Mean +- Standard deviation, the part is penalized. The overall fitness, F(x) for a given individual x reads as:

$$F(x) = \sum_{i=1}^{13} \alpha_i F_i(x)$$

; where α_i are the weights used to prioritize the various objectives (the actual values are out of the scope of this paper) and F_i are the 13 objective functions described above.

Deriving the proper fitness functions was accomplished by comparing ratings of randomly generated form-states with ratings of the actual corpus. As individuals could be penalized for falling outside of a single standard deviation, outlying parts within the corpus would still be penalized. Some objectives beyond the thirteen described here were not used, despite their apparent usefulness. For example, comparing consistency within parts between section repetitions inside a track: while this seemed like a good musical aspect to test, no discernible difference could be found between the random generation and the corpus analysis, due to the wide deviation found within the corpus itself.

Finally, an additional fitness adjustment was added to penalize null phrases – those that are completely silent.

A tournament selection (of size two) is then made within the population of 100 individuals: for each tournament, winners (with duplicates removed) automatically pass to the next generation. This elitist selection is repeated until half of the next generation is generated. Individuals that win no tournaments are replaced with new individuals that are mated from parents that are both winning individuals. Mating is done using multi-point crossover: two randomly chosen parents' complete sections are randomly combined, resulting in a single offspring.

A maximum of 20 generations are evolved, with a potential exit if the highest rated individual's score drops below a threshold. The threshold is set as the mean coefficient for each fitness criteria derived from the corpus analysis: once the evolution passes the evaluated state of the corpus, evolution is complete.

3.2.3 Determining Pattern Change Locations

The fitness functions within the GA for the form-state generation are primarily concerned with sectional relationships through densities, as well as relationships between parts. The progression of individual parts via the introduction of new patterns is not considered. How patterns are distributed within a part is now discussed.

Although the maximum number of patterns within a part could be derived through probabilities, a heuristic solution as to *when* and *where* these patterns should change was not apparent; therefore, more analysis of the corpus was required. First, the probability of a particular section having a pattern change contained within it was determined (see Table 2). A higher percentile of pattern changes occur in the Lead-in (A), as this section is generally used to introduce important parts. The remaining parts are often introduced in the Intro (B) section: in fact, it is the introduction of specific parts – such as the bass – that often will determine sectional divisions. The Verse (C) has to lowest amount of new pattern introductions, as it tends to be a section in which previously introduced individual patterns are combined. Finally, the Breakdown (D) often introduces new patterns in the synthesizer parts, while the drums and percussion drop out.

Table 2. Probability of new beat patterns occurring in a section, for the "Breaks" corpus

A	0.4
B	0.15
C	0.04
D	0.12
E	0.07

A different method was used in generating the number of patterns needed per section, rather than a straightforward stochastic probability generation. It was found that there was a tremendous variation between tracks for the number of parts in a section, and using stochastic methods would tend to flatten out this data; instead, the actual values from the corpus are stored, and a random selection is made from these values: as such, we are sampling the corpus data directly.

Table 3. Probability distributions for new patterns occurring in the first phrase of a section, in the second through penultimate phrases, or the last phrase, "Breaks" corpus.

	First phrase	*Middle phrases*	*Final phrase*
A	0.9	0.1	0.0
B	0.85	0.05	0.1
C	0.93	0.03	0.04
D	0.72	0.07	0.21
E	0.415	0.17	0.415

Once the total number of patterns within a track is distributed through the sections, the *location* of the pattern changes is determined within the section's phrases. An array is calculated accumulating the number of pattern changes occurring in the corpus during a section's first phrase, the second through

penultimate phrases, and the final phrase (see Table 3). Given the number of pattern changes that are required for the section, a roulette-wheel selection is made from the percentile to determine their location(s).

3.2.4 A Population of Strong Individual Form-States

Once the system was put into artistic practice, it became obvious that users desired some control over the generation of form. Rather than being forced to interact with the system in a Monte-Carlo fashion – generate, accept the result, or generate again – it was found more useful to generate a large population of strong individuals, each the strongest of a separate evolution, and select from those based upon user-set conditions. Once this large (>2000) population is generated (which takes several hours), a user can select preferred parameter states – for example, the presence of two melodic synth parts, or the presence of the auxiliary percussion – and the system returns the individual form-state that is rated closest, using a multi-dimensional Euclidian distance function (see Figure 4).

Figure 4. Setting targets for selection from a population of form-states. As left, light-grey displays ranges from the population, green lines display mean values, dark grey show user selection. At right, weightings for selections are determined: in this case, only density, MSynth1 & 2, and Keys rating are used.

4. PATTERN GENERATION AND SELECTION

4.1 Beat Pattern Generation

Pattern generation involves methods that cannot be described in detail here, for reasons of space. Only a cursory description is given here, while a more thorough discussion can be found in a companion paper [11]. Three different methods are used to generate beat patterns, including:

1. Generating individual subparts (Kick, Snare, Closed Hihat, and Open Hihat) via probabilities for a given onset occurring at a given time (beat);
2. First-order Markov generation of individual subparts;
3. First-order Markov generation of combined subparts.

In the first case, probabilities are calculated for each subpart (see Figure 5).

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. Continuations are considered across eight-bar phrases. As there is a great deal of repetition, certain patterns are clearly emphasized within the Markov table.

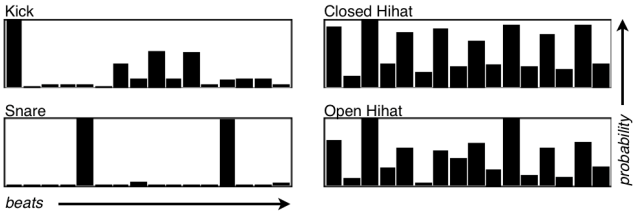

Figure 5. Onset probabilities for individual subparts, 1 bar (sixteenth-note subdivisions), C section, "Breaks" corpus

In the third case, data is stored by beat as a 4 bit flag: bit 4 = Kick; bit 3 = Snare; bit 2 = Closed Hihat; bit 1 = Open Hihat. This method ensures that polyphonic relationships between parts are encoded, as well as time-based relationships. This 4-bit representation of beat patterns is used not only for drum patterns, but the three auxiliary percussion parts as well (see Figure 6).

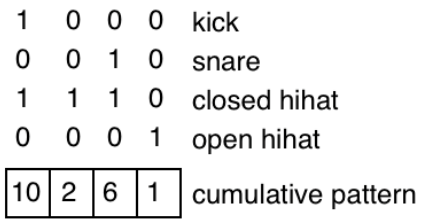

Figure 6. Representing the 4 drum subparts as 4-bit cumulative values for one beat (sixteenth-note subdivision).

In all cases, the data used for generation is context dependent: separate databases are maintained for separate sections. Thus, when a beat pattern is required that first appears in section B, only those patterns that appeared in section B of the corpus are used. Note that this example describes the generation of beat patterns; similar methods are used for the generation of all parts, including pitched parts such as bass and synthesizers.

4.2 Pattern Selection from a Population

In actual usage, it was found that the third method produced the most accurate re-creations of beat patterns found in the corpus, yet the first method produced the most surprising, while maintaining usability. Rather than selecting only a single method for beat generation, it was decided that the three separate methods provided distinct "flavors"; as such, all three methods were used in the generation of a large (>2000) database of potential patterns.

As a track's beat pattern influence the "feel" of the track, the user can select general tendencies for the pattern as targets (see Figure 7).

For beat generation, these are separated between the kick/snare, and hihats, and include:

– Density: a coefficient for the number of actual onsets over the pattern divided by the possible number of onsets;

– Syncopation: the number of onsets that are not on the part's usual strong beats;

– Symmetry: the number of onsets in the second half of the pattern that occur in the same location in the first half of the pattern.

What we have found is that rather than attempting to influence generation (which often led to continually generating and rejecting patterns until an "interesting" pattern appeared), the population of strong individuals can be created that contains a great deal of diversity. The user can fine-tune the criteria in order to influence the final selection.

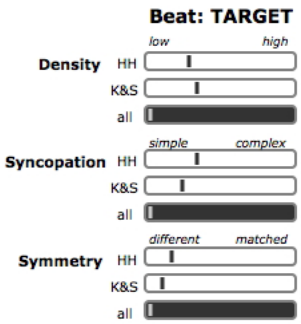

Figure 7. User-set targets for selection from pattern population.

Furthermore, target selection from a population provides an additional benefit in the next stage of our research: successive generations of entire compositions can be guaranteed to be divergent by ensuring targets for parameters, such as beat patterns, are different for each generation.

5. CONCLUSIONS AND FUTURE WORK

We have described our ongoing investigation into generating EDM using population-based evolutionary methods, based upon expert analysis and transcription of a corpus. We described a method of generating and selecting overall forms, as well as individual patterns, through both evolutionary and probabilistic methods. Throughout this work, constraints and heuristics have been used to maintain vertical and horizontal relationships as found in the corpus. Note that our implementation of generative methods for one-hits and fills, both integral elements of variety in EDM, have not been discussed for reasons of brevity.

Future work includes expanding the potential targets for selection from the populations, based upon user requests. Through extended interaction with the system, users will fine-tune their targets, rather than simply re-generating individuals hoping for a successful result.

Two essential elements of EDM have not been discussed: (1) the timbral selection of parts (i.e., instruments) and dynamic signal processing (i.e. audio mixing and post-production techniques). Although we are generating audio [24], the final product still relies upon human selection for these aspects, both of which are actively being researched.

Lastly, if one considers that the described system is a virtual EDM Producer, we plan to develop a matching virtual DJ, one that assembles existing tracks created by the Producer into hour-long sets. Assemblage would involve signal analysis of every generated track's audio in order to determine critical audio features; individual track selection would then be carried out based upon a distance function between the track data and a generated timeline, which may or may not be derived from analysis of a given corpus consisting of DJ sets. This timeline could be varied in performance based upon real-time data: for example, movement analysis of the dance-floor could determine the ongoing success of the selected tracks.

Our evaluation plan includes submitting our audio results to live situations, such as upcoming algoraves[3] – the system has been accepted for performance at two festivals of Generative Art. Audio output from our system can be found online [24].

[3] http://algorave.com/

6. 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.

7. REFERENCES

[1] Beyls, P. 2009. Interactive Composing as the Expression of Autonomous Machine Motivations. In: *Proceedings of the International Computer Music Conference* (Montreal, Canada, August 2009) 267–74.

[2] Biles, J. 2001. Autonomous GenJam: Eliminating the Fitness Bottleneck by Eliminating Fitness. In: *Proceedings of the 2001 Genetic and Evolutionary Computation Conference Workshop Program*, San Francisco.

[3] Blackwell, T, and Young, M. 2004. Swarm Granulator. Applications of Evolutionary Computing. In: *Proceedings of EvoWorkshops 2004*. Springer-Verlag, 399–408.

[4] Butler, M. 2006. *Unlocking the Groove: Rhythm, Meter, and Musical Design in Electronic Dance Music.* Bloomington, Indiana University Press.

[5] Cope, D. 2005. *Computer Models of Musical Creativity.* Cambridge, MA: MIT Press.

[6] 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 Confence (ISMIR)*, Kobe, 465–469.

[7] Eigenfeldt, A., and Pasquier, P. 2012a. 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.

[8] Eigenfeldt, A., and Pasquier, P. 2012b. Creative Agents, Curatorial Agents, and Human-Agent Interaction in Coming Together. In: *Proceedings of the 9th Sound and Music Computing Conference*, Copenhagen, 181–186.

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

[10] Eigenfeldt, A. 2012. "Corpus-based recombinant composition using a genetic algorithm" *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 16(12), 2049–2056.

[11] Eigenfeldt, A., and Pasquier, P. 2013. Considering Vertical and Horizontal Context in Corpus-based Generative Electronic Dance Music. Submitted to: *International Conference of Computational Creativity*, Sydney.

[12] Goldberg, D. 1989. *Genetic Algorithms in Search, Optimization and Machine Learning.* Reading, MA: Addison-Wesley Professional.

[13] Kaliakatsos-Papakostas, M., Floros, A., Kanellopoulos, N., and Vrahatis, M.N. 2012. Genetic Evolution of L and FL-systems for the Production of Rhythmic Sequences. In: *Proceedings of the 2nd Workshop in Evolutionary Music held during the 21st International Conference on Genetic Algorithms and the 17th Annual Genetic Programming Conference (GP) (GECCO 2012)*, 461–468.

[14] Kaliakatsos-Papakostas, M., Floros, A., and Vrahatis, M.N. 2013. EvoDrummer: Deriving rhythmic patterns through

interactive genetic algorithms. In: *Proceedings of EvoMusArt Conference 2013*, Vienna (to appear).

[15] MacCallum, R., Mauch, M., Burt, A., and Leroi, A. 2012. Evolution of music by public choice. In: *Proceedings of the National Academy of Sciences*, 109(30), 12081–12086.

[16] McCormack, J. 2001. Eden: An Evolutionary Sonic Eco-System. In: *Advances In Artificial Life*, LNCS. Springer, 133–142.

[17] Miranda, E., and J. Biles, eds. 2007. *Evolutionary Computer Music*. London: Springer.

[18] Thywissen, K. 1996. GeNotator: An environment for investigation the application of genetic algorithms in computer assisted composition. In: *Proceedings of the 1996 International Computer Music Conference*, San Francisco, 274–277.

[19] Todd, P., Werner, G.1999. Frankensteinian methods for evolutionary music composition. In: *Musical networks: Parallel distributed perception and performance*, Griffith, N., Todd, P., eds., Cambridge, MA: MIT Press/Bradford Books, 313–339.

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

[21] Weinberg, G., Godfrey, M., Rae, A., and Rhoads, J. 2008. A Real-Time Genetic Algorithm in Human-Robot Musical Improvisation. In: *Computer Music Modeling and Retrieval, Sense of Sounds*, Springer, Berlin, 351–359.

[22] Whitelaw, M. 2004. Metacreation. Art and Artificial Life. Cambridge, MA: MIT Press.

[23] Wooller, R., and Brown, A. 2009. TraSe algorithm: automatic evolutionary morphing of electronic dance music. In: *Improvisation: Australasian Computer Music Conference*. Australasian Computer Music Association, 64–71.

[24] http://soundcloud.com/loadbang. Accessed April 8, 2013.