Journal of New Music Research

ISSN: 0929-8215 (Print) 1744-5027 (Online) Journal homepage: http://www.tandfonline.com/loi/nnmr20

Automatic Synthesizer Preset Generation with PresetGen

Kıvanç Tatar, Matthieu Macret & Philippe Pasquier

To cite this article: Kıvanç Tatar, Matthieu Macret & Philippe Pasquier (2016) Automatic Synthesizer Preset Generation with PresetGen, Journal of New Music Research, 45:2, 124-144, DOI: 10.1080/09298215.2016.1175481

To link to this article: http://dx.doi.org/10.1080/09298215.2016.1175481

Published online: 01 May 2016.

Submit your article to this journal

Article views: 203

View related articles

View Crossmark data

Full Terms & Conditions of access and use can be found at
http://www.tandfonline.com/action/journalInformation?journalCode=nnmr20

Download by: [Simon Fraser University] **Date:** 29 July 2016, At: 13:21

Journal of New Music Research, 2016
Vol. 45, No. 2, 124–144, http://dx.doi.org/10.1080/09298215.2016.1175481

Routledge
Taylor & Francis Group

Automatic Synthesizer Preset Generation with *PresetGen*

Kıvanç Tatar, Matthieu Macret and Philippe Pasquier

Simon Fraser University, Vancouver, Canada

(Received 12 September 2015; accepted 24 March 2016)

Downloaded by [Simon Fraser University] at 13:21 29 July 2016

Abstract

We refer the task of finding preset(s) (i.e. set(s) of synthesizer parameters) that approximates a target sound best, as the preset generation problem. *PresetGen* addresses this problem regarding the real world synthesizer, OP-1. The OP-1 consists of several synthesis blocks, and it is not fully deterministic. We propose and evaluate a solution to preset generation using a multi-objective Non-dominated Sorting-Genetic-Algorithm-II. *PresetGen* handles the full problem complexity and returns a small set of presets that approximate the target sound best by covering the Pareto front of this multi-objective optimization problem. Moreover, we present an empirical evaluation experiment that compares the performance of three human sound designers to that of *PresetGen*. The results show that *PresetGen* is human-competitive.

Keywords: sound synthesis, machine learning, audio analysis, instruments

1. Introduction

Exploration of a synthesizer's sound space requires theoretical and empirical expert knowledge. Composers have to abandon making music to concentrate on the task of programming a synthesizer, i.e. tuning parameters to create the desired sound. Regarding a complex synthesizer, the size of the parameter search space can quickly become large and challenging for users to handle manually. Preset generation to approximate a target sound, even in a principled fashion, can become a time-consuming activity.

Synthesizer manufacturers often provide the user with a large number of parameter settings, also called presets. Presets are starting points for users to explore the synthesis sound space. However, even if these presets are meant to convey the variety of sounds that the engine is capable of generating, they fail to cover the entire synthesis sound space. In this work, we focus on the problem of synthesizer preset generation which consists in finding the preset that approximates a target sound best. We aim to provide an automatic system, *PresetGen*, to users so that they can generate their own presets for their target sounds without dealing with complex parameter settings.

This work examines the use of Evolutionary Computation (EC) for the preset generation problems of a modern commercial synthesizer, the OP-1. The OP-1 (developed by Teenage engineering (n.d.)) is an all-in-one portable synthesizer, sampler and controller. The OP-1 introduces additional challenges compared to previously studied synthesizers (Bozkurt & Yüksel, 2011; Horner & Beauchamp, 1996; Horner, Beauchamp, & Haken, 1993; Lai, Jeng, Liu, & Liu, 2006; Mitchell, 2012; Riionheimo & Välimäki, 2003; Schatter, Züger, & Nitschke, 2005; Vuori & Välimäki, 1993). OP-1 contains several synthesis engines, effects (FX) and Low-Frequency-Oscillators (LFOs), which make the parameter search space larger and complex. Furthermore, the OP-1 is not fully deterministic, therefore, the sound generated for a given preset is slightly different at each generation.

In this study, we base our evaluation design on Johnson's (2002) recommendation about the experimental analysis of algorithms. We focus on ensuring reproducibility and comparability. Moreover, we use two types of sound similarity: computational sound similarity and perceptual sound similarity. *PresetGen* uses computational sound similarity to generate presets that approximates a target sound (see Section 5.4). We focus on perceptual sound similarity in our empirical evaluation experiment (see Section 7).

Section 2 presents a general background on EC and explains a variation of the canonical Genetic Algorithms (GA), called the Non-dominated Sorting Genetic Algorithm- II (NSGA-II). Section 3 provides a literature review of previous works on applications of EC to solve the problem of synthesizer preset generation with various synthesis techniques. Section 4 describes the OP-1 synthesizer and analyses the problem of generating synthesizer presets that match a given target

Correspondence: Kıvanç Tatar, School of Interactive Arts and Technology, 250 - 13450 - 102nd Avenue, Surrey, British Columbia, V3T 0A3 Canada. E-mail: ktatar@sfu.ca

© 2016 Informa UK Limited, trading as Taylor & Francis Group

Downloaded by [Simon Fraser University] at 13:21 29 July 2016

sound. We also emphasize the difficulties that are unique to the OP-1 synthesizer. We describe *PresetGen*'s system design in Section 5. Section 5.1 presents the methodology we designed to solve the preset generation problem.

Section 6.1 describes the sound collection that we used to study the performance statistics of *PresetGen*. We explain why we chose certain 24 sounds in the two categories of contrived and non-contrived sounds. The term contrived sounds refers to the target sounds generated by the synthesizer itself; in this case, OP-1. Twelve of these sounds were contrived sounds, the other 12 were non-contrived sounds. Section 6.2 shows the correlation between *PresetGen*'s fitness function and the distance between target preset and the matching preset for a contrived sound. Section 6.3 presents *PresetGen*'s performance statistics by running *PresetGen* 10 times for each target sound. First, we use contrived target sounds to assess the ability of *PresetGen* to retrieve the target synthesizer's parameters. We study *PresetGen*'s performance on Fast Fourier Transform (FFT), Envelope and Short-Time Fourier Transform (STFT) objectives with the results of the contrived sound experiments. Second, we present the performance statistics of *PresetGen* with non-contrived sounds. Section 7 presents the empirical evaluation of OP-1 preset generation for non-contrived sounds. In the empirical evaluation experiment, we study if *PresetGen* can improve a preset generation task of non-contrived sounds regarding the quality and time complexity.

2. Background on evolutionary computation

2.1 Background on genetic algorithms

Inspired by Darwinian evolution, EC is a family of parallel search algorithms that solve computational optimization problems. EC employs the ideas of *survival of the fittest*, *individuals* and *populations*, *genetic inheritance*, *crossover*, *mutation*, *reproduction*, *recombination*, *elitism selection* and *generation*. EC uses phenotype and genotype dichotomy to represent a real-world phenomenon (Sivanandam & Deepa, 2007). Genetic Algorithm (GA) is a branch of EC, introduced by Holland (1975).

In GA abstractions, an individual (genotype) is a candidate solution (phenotype) to a given problem. A population is a set of individuals (solutions). GAs represent an individual as bitstrings or set of numbers. GAs assign a fitness value to each individual using a *fitness function*. These fitness values indicate the performance of individuals (solutions) for the problem of interest. Each generation, *crossover* and *mutation* genetic operators create new individuals. *Selection* and *reproduction* genetic operators ensure that GAs keep individuals with higher performance of solving the problem. Moreover, GAs implement *elitism* to preserve the genotype diversity in the population (Sivanandam & Deepa, 2007). GAs continue the search for the optimum solution until the algorithm reaches a stopping criteria, such as maximum number of generations or a user defined fitness value.

2.2 Multi-objective genetic algorithms

Multi-objective GAs (MO-GAs) optimize multiple objective functions simultaneously. Comparing two individuals in an MO-GA population, an individual *dominates* another solution if it is better for one or more fitness objectives without being worse for the remaining fitness objectives. MO-GAs return a set of *Pareto-optimal* solutions. In a set of *Pareto-optimal* solutions, a solution cannot dominate another solution. The *Pareto front* is the set of solutions that are *Pareto-optimal* at a given moment. The *cumulative Pareto front* includes all the Pareto-optimal solutions encountered in an MO-GA run[1].

In this study, we implement an MO-GA, called NSGA-II (Deb, Pratap, Agarwal, & Meyarivan, 2002). NSGA-II has become the standard approach solving multi-objective problems. In the next subsections, we define the notion of non-dominated sorting and crowding distance, and describe how they are used in the main loop of the NSGA-II to converge toward an optimized Pareto front.

2.2.1 A non-dominated sorting approach

We divide an MO-GA population into non-dominated fronts (subsets) in which individuals do not dominate each other. NSGA-II sorts the population in non-dominated fronts for each generation.

Let's consider a population of size N and M objectives. First, the algorithm compares each individual with every other individual in the population to find if it is dominated. Then, the non-dominated individuals constitute the first non-dominated front. We temporarily exclude the individuals in the first non-dominated front and repeat the above procedure to find the second non-dominated front. The algorithm uses the same procedure for finding third and higher levels of non-dominated fronts. The first non-dominated front is the Pareto front. Each generation, *PresetGen* adds the individuals in the Pareto front to the cumulative Pareto front.

Regarding the computational complexity, the worst case is when there are N fronts and one solution in each front. To sort the population in non-dominated fronts, each individual is compared with others to find if it is dominated. This procedure requires $\mathcal{O}(MN)$ computations. Then, this computation is repeated for all individuals in the population. This requires $\mathcal{O}(MN^2)$ computations. Then, there are N fronts in the worst case. Hence, the computational complexity is $\mathcal{O}(MN^3)$. NSGA-II uses a fast non-dominated sorting approach. Considering an individual (p) in an MO-GA population, this approach assigns a non-domination level (n_p) and a set of dominated individuals (S_p) to each individual in the population. Using n_p and S_p, NSGA-II requires $\mathcal{O}(MN^2)$ computations (Deb et al., 2002).

[1]The cumulative Pareto front is referred as *Pareto front Hall of Fame* in Distributed Evolutionary Algorithms in Python (DEAP) NSGA-II implementation (Fortin, De Rainville, Gardner, Parizeau, & Gagné, 2012) (see Section 5.7).

Downloaded by [Simon Fraser University] at 13:21 29 July 2016

2.2.2 Diversity preservation

Along with convergence to the *Pareto-optimal* set, we design *PresetGen* to maintain a high spread of diversity in the solution set. NSGA-II assigns to every individual a *crowding distance* (Deb et al., 2002). Crowding distance calculation is as follows:

(1) Sort the population along each fitness objective. We end up with different rankings of the same population for each fitness objective. x_{ij} is the jth objective value of the ith individual in the population sorted along the jth objective.
(2) Assign an infinite distance value to the boundary solutions—the individuals with the smallest and the largest fitness value along that objective.
(3) Calculate the crowding distance of the remaining individuals. For each fitness objective, $d_j(x_{(i-1)j}, x_{ij})$ represents the Euclidian distance between the jth objective fitness values of individuals $i-1$ and i. Calculate the ith individual's crowding distance along the jth objective with the formula,

$$\frac{|d_j(x_{(i-1)j}, x_{ij}) - d_j(x_{ij}, x_{(i+1)j})|}{f_{j,max} - f_{j,min}}, \quad (1)$$

where $f_{j,max}$ and $f_{j,min}$ refers to the corresponding objective's maximum and minimum fitness value in the population.
(4) Assign a crowding distance to each individual by summing an individual's crowding distances along each fitness objective.

Equation 2 summarizes the crowding distance calculus for an individual in the population.

$$d(x_i) = \sum_{j=1}^{N_{obj}} \frac{|d_j(x_{(i-1)j}, x_{ij}) - d_j(x_{ij}, x_{(i+1)j})|}{f_{j,max} - f_{j,min}}, \quad (2)$$

where N_{obj} is the number of objectives.

The introduction of crowding distance provides that the boundary solutions are more likely to be kept in the next generation (see following section for more details).

2.2.3 NSGA-II main loop

We present the NSGA-II procedure in Figure 1 First, a population P_0 of size n_{pop} is randomly initialized. Then, the fitnesses of all individuals are evaluated using the fitness function. NSGA-II sorts the population using *non-domination levels* assigned to each individual. Using the algorithm described in Section 2.2.2, a *crowding distance* is also calculated for each individual.

Contrary to the canonical GA, NSGA-II uses a tournament selection operator based on the dominance between two individuals. If none of the two individuals dominate the other, the selection is made based on the crowding distance. NSGA-II prioritizes individuals with higher crowding distance. The recombination and mutation operators are used to create an

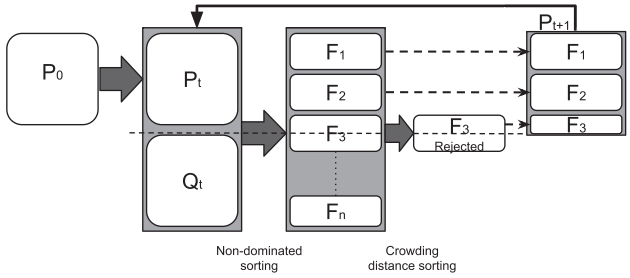

Fig. 1. NSGA-II procedure (Deb et al., 2002).

offspring population Q_t of size n_{pop}. A combined population of $P_t \cup Q_t$ is considered. This population is sorted according to non-domination. Since all previous and current population individuals are included in $P_t \cup Q_t$, *elitism* is ensured. Solutions belonging to the best non-dominated set F_1 are from the combined population. F_1 is passed to the next generation with the highest priority. If the size of F_1 is smaller than n_{pop}, we definitely choose all members of the set for the new population P_{t+1}. The remaining members of the new population (P_{t+1}) are chosen from subsequent non-dominated fronts in the order of their ranking. This procedure is continued until no more sets can be completely included. Say that the set F_n is the last non-dominated set beyond which no other set can be accommodated. The count of solutions in all sets from F_1 to F_n is larger than the population size n_{pop}. To choose exactly n_{pop} population members, we sort the solutions of the last front by *crowding distance* in decreasing order and choose the best solutions needed to obtain a population size of n_{pop}.

3. Background on evolutionary computation for sound synthesis

Regarding the applications of EC in sound synthesis, we start our literature review with optimization problems of relatively low complexity to more general problems of higher complexity, involving modern synthesizers that embed multiple complex synthesis engines.

The complexity of the preset generation problem can vary tremendously according to the number and nature of the synthesis parameters to search. First efforts focused on optimizing a limited number of synthesis parameters. Horner and Beauchamp (1996) studied preset generation with additive synthesis. They used GA to find how many breakpoints they needed to match a target sound on their piecewise-linear approximation of additive synthesis amplitude and frequency envelopes. The number of oscillators to use and their frequencies were not determined by the GA, but through spectral analysis.

Chan, Yuen, and Horner (1996) implemented GA with the discrete summation and hybrid sampling wavetable model synthesis method to match a musical instrument tone. The hybrid sampling wavetable method uses sampling for the attack of a sound, and wavetable synthesis to gradually change

Downloaded by [Simon Fraser University] at 13:21 29 July 2016

sustain and release. This method can synthesize sounds with a dynamic spectrum using multiple wavetables. Chan et al. (1996) used this synthesis method as a spectral interpolation approach on the problem of preset generation. GA is implemented to determine the basis spectra and the best amplitude envelope for the spectral interpolation.

Wakefield and Mrozek (1996) used subtractive synthesis to create artificial reverberation. A GA was used to search for low-order filter parameters so that the generated impulse response best matched that of a target room transfer function.

Horner et al. (1993) used a GA to optimize several frequency modulation (FM) synthesis parameters: the modulation indices, carrier and modulator frequencies. The spectral error between the original and target spectra served as a fitness function in guiding the GA's search for the best FM parameters to mimic instrumental sounds. In our previous work, we used a similar technique to optimize modulation indices, carrier and modulator frequencies for Modified FM (ModFM) synthesis (Macret, Pasquier, & Smyth, 2012).

Vuori and Välimäki (1993) applied a GA to estimate parameters of a non-linear physical flute modelling synthesis. The genotype of this system presented eight different parameters of the physical model. The spectral error between the original and matched spectra served as the fitness function. The authors reported that the algorithm converged smoothly and effectively towards the target sound.

Bozkurt and Yüksel (2011) conducted synthesizer preset generation experiments with GAs in application to multiple-modulator FM synthesis. Contrary to the FM synthesis systems previously presented (Horner et al., 1993; Macret et al., 2012), their GA implementation included all parameters that control their FM synthesis method.

Mitchell (2012) compared three algorithms—Multi-member Evolution Strategy (such as 1+4), Multi-start (1+1) Evolution Strategy and Clustering Evolutionary Strategy (CES)—to generate presets for FM synthesizers. CES clusters the population at the beginning of each iteration using k-means clustering. This parent population re-clustering ensures that convergences on the same niche merge to form a single cluster. Mitchell (2012) stated that this was beneficial in multi-modal search spaces (see Section 4). The synthesis architecture consists of parallel FM synthesis blocks. The number of these blocks was also another parameter to be optimized. This implementation used Short-Time Fourier Transform (STFT) in the fitness function. Mitchell (2012) divided the evaluation of these algorithms into two: non-changing static tones and time-varying dynamic sounds. Considering these three algorithms, Mitchell (2012) concluded that CES performed the best in the experiments.

Yee-King and Roth (2008) used a GA to generate presets of Virtual Studio Technology instruments (VSTi) to match a given target sound. This implementation did not include VST FXs. The system is called *SynthBot*. Although VSTi include more complex synthesis architectures than former studies, the OP-1's particular synthesis architecture introduces more complexity to the preset generation problem. We explain these new challenges in Section 4. *SynthBot* uses single objective of the sum squared error between the target sound's and the matching sound's Mel Frequency Cepstrum Coefficients (MFCCs) in its fitness function. Trials show that the single objective approach gives unsatisfactory results in the case of OP-1. We present results of our final multi-objective design in Section 5.

EC methods are also implemented to evolve synthesizer architectures in conjunction with the synthesizer presets (Garcia, 2001; Macret & Pasquier, 2014; Takala, Hahn, Gritz, Geigel, & Lee, 1993; Wehn, 1998). These studies implement GA, Genetic Programming (GP) and Coevolutionary Genetic Programming. However, the evolution of synthesis architectures is out of the scope of this paper.

4. The OP-1 synthesizer

The OP-1 is an all-in-one portable synthesizer, sampler and controller developed by Teenage Engineering (TE) (n.d.) (Figure 2). Figure 3 shows an overview of the OP-1's synthesis architecture. TE provided us with a C++ library that embeds the functionalities of the OP-1. We had access to seven different sound synthesis engines (FM, Digital, DrWave, String, Cluster, Pulse and Phase), four different FX (Delay, Grid, Punch, Spring) and three different LFOs (Tremolo, Value, Element). In the following, the parameters selecting the engine, FX and LFO will be referred to as *type parameters*. Only one engine, one effect and one LFO can be used at a given time to produce a sound. An ADSR envelope is also always applied to the sound. Once chosen, the synthesizer engine, FX, LFO and ADSR can each be controlled individually by four parameters, using four colour-coded knobs. We refer the parameters controlling the knobs as *knob parameters*. The knob parameters are mapped to integers ranging from a minimum of 0 to a maximum of 32,767, corresponding to the fine-tuning mode of the OP-1. The OP-1 has 24 physical keys and it is possible to change the octave from -4 to 4. Therefore, 120 different keyboard keys ($8 \times 12 + 24$) are available. We refer to a set of OP-1 *knob parameters* and OP-1 *type parameters* as an OP-1 preset. More details about the OP-1 can be found on the Teenage engineering (n.d.) website.

Equation 3 gives the number of different possible presets.

$$N_{eng} \times N_{LFO} \times N_{FX} \times N_k^{N_{knobs} \times N_t} \times N_{keys}, \qquad (3)$$

where N_{eng} is the number of engine types, N_{LFO} is the number of LFO types, N_{FX} is the number of FX type, N_t is the number of modules that can be controlled by knobs (engine, LFO, FX and ADSR), N_k is the number of possible integer values for each knob and N_{keys} is the number of keys. Their numerical values are given in Table 1 Then, an estimate of the total number of possible combinations for the OP-1 synthesizer is 10^{76}.

Searching the synthesizer parameters space to generate a preset that can match a given target sound has all the characteristics of a real-world problem. First, the search is very large (10^{76} possible different combinations). By comparison, the number of atoms in the observable universe is estimated at

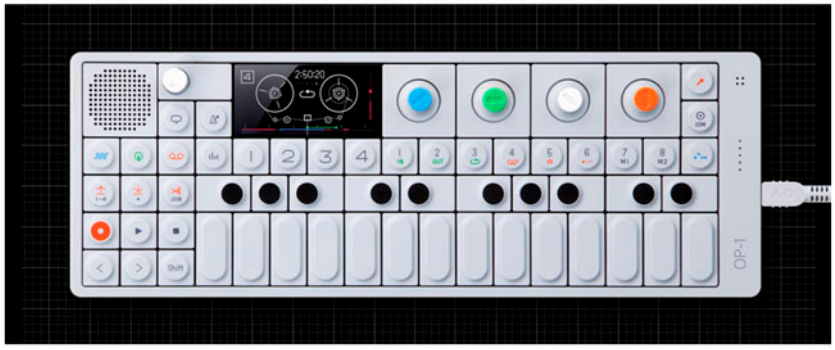

Fig. 2. The OP-1 synthesizer by Teenage Engineering.

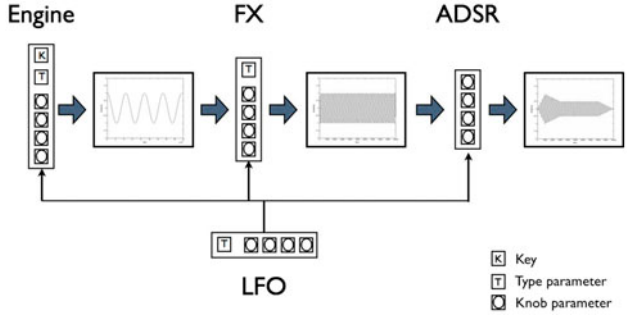

Fig. 3. The OP-1's modular synthesis architecture.

Table 1. Synthesizer parameters complexity.

N_{eng}	N_{LFO}	N_{FX}	N_k	N_{knobs}	N_t	N_{keys}
7	3	4	32767	4	4	120

10^{80}. Second, the OP-1 synthesizer is not fully deterministic. The output sound is slightly different each note generation, which induces noise in the evaluation. This non-determinism can slow down or even mislead the search. We conducted two experiments to show the non-determinism of the OP-1 and results are available online (Tatar, Macret, & Pasquier, 2015). Third, the search space is multi-modal. For example, switching from one engine to another completely changes the nature of the output sound. As a result, fitness values substantially change, causing a discontinuity in the fitness landscape. This multi-modality also modifies the mapping of the *knob parameters*. For example, the *knob parameters* for the *FM* engine do not map to the same synthesis parameters as the *knobs parameters* for the *Digital* engine. Trials showed that there are a large number of local minima (see Section 6.2.1). For instance, it is often possible to get a similar level of sound approximation using two different engines. Given these problem characteristics, it is not conceivable to use a simple optimization technique such as hill climbing or greedy algorithm to find a good set of parameters to match a given target sound. These techniques are highly dependent on the initial conditions and do not scale well to large and difficult search spaces (Roth, 2011).

5. System design

5.1 Methodology

As described in Section 2.2, GAs are especially well adapted to the characteristics of our problem. First, GAs scale very well to the large and complex search space induced by the OP-1. Contrary to gradient search methods, they are less susceptible to converge prematurely to a local optimum (Rocha & Neves, 1999).

Second, GAs also perform well in search spaces where the evaluation is approximative or noisy (Jin & Branke, 2005), as is the case with the OP-1 and its non-fully deterministic output. Adjustable selection pressure makes it possible to keep diversity in the population. A large number of individuals are evaluated for each generation. Because mutation and crossover are stochastic operators, it is common for an individual to be rediscovered several times during the evolution. The fact that *PresetGen* re-evaluates and rediscovers an individual, reduces the effect of the noise in the evaluation due to the non-determinism of OP-1.

GAs are complex algorithms with a large set of parameters to tune (population size, stopping criteria, choice of the genetic operators). We explored several options to find the best configuration for the GA. In the following sections, we refer to the target sounds generated using the OP-1 as contrived sounds (Mitchell & Creasey, 2007). Using contrived sounds as target sounds has two advantages. First, it ensures that a solution exists. Second, we can measure the performance of the algorithm by calculating its distance to the optimal solution.

5.2 Representation of OP-1 presets

PresentGen implements a string of 257 bits as the chromosome representing an OP-1 preset. *PresetGen* uses the Gray code to encode the OP-1 parameters. Barbulescu, Watson, and Whitley (2000) show that Gray code representation has benefits to escape local minima for parameter optimization problems. In the Gray code representation, two successive values differ by only one bit. The number of distinctly possible bit strings is then $2^{257} = 10^{257\log_{10}(2)} \approx 10^{77}$. In Section

Downloaded by [Simon Fraser University] at 13:21 29 July 2016

Downloaded by [Simon Fraser University] at 13:21 29 July 2016

4, we calculated that the number of distinct possible OP-1 presets have an order of magnitude of 10^{76}. The difference (a factor of 10) between these two orders of magnitude is as follows. When the number of values to encode is not power of 2, the binary encoding encodes for more values than necessary. For example, we have to encode 120 different keys in our chromosome (see Table 1). With 6 bits, it is possible to encode $2^6 = 64$ different keys and with 7 bits, $2^7 = 128$ different keys. Then, we chose to use 7 bits and applied a scaling function to keep the decoded integers between 0 and 119. Hence, this difference of 8 between the number of keys to encode and the number of possible distinct bit strings (using 7 bits) causes the difference in the orders of magnitude.

5.3 Genetic operators

Losing diversity during the evolution is a normal phenomenon given that we apply a selection pressure on the population. However, a lack of diversity can lead to premature convergence because there is not enough genetic material to explore the fitness landscape. One reason for the loss of diversity is the recombination of identical chromosomes. Indeed, when a given chromosome is selected twice for crossover, two offsprings identical to their parents are produced. This phenomenon causes the diversity to go down. To avoid this situation, we apply a crossover operator that tests the parent chromosomes before recombining them. If they are identical, the first offspring will be a copy of the parents and the second offspring will be a new randomly generated chromosome. This simple technique is shown to be efficient in slowing down the diversity loss and prevent premature convergence (Rocha & Neves, 1999). *PresetGen* uses a two-point crossover and a crossover rate of 60%.[2]

The mutation operator participates in both exploration and exploitation (local search). Flipping one bit in a Gray code can either lead to a small change in the coded parameter (local search) or a relatively large change in the coded parameter (exploration, by jumping to another area of the fitness landscape). Table 2 shows two examples of mutation. Flipping one bit in the Gray code can either lead to a single increment in the integer value (4095 to 4096), or lead to a large change in the integer value (4095 to 2048). This flip-bit mutation operator is applied to every individual in the population, whether crossover is applied or not. In our system, the probability of flipping k bits in an N_{bits} long chromosome follows a binomial law with $p = 1/N_{\text{bits}}$ and $n = N_{\text{bits}}$ (Deb et al., 2002).

The population's diversity is critical for the success of the GA. We explain how NSGA-II preserves the diversity in Section 2.2. Furthermore, we use the following approaches to measure the diversity:

[2]Crossover rate is the probability of the *crossover* operator to be applied on two individuals during *recombination* (Sivanandam & Deepa, 2007).

Table 2. Results for the benchmark.

Integer	Binary	Gray
4095	0111111111111	0100000000000
4096	1000000000000	**1**100000000000
2048	0100000000000	01**1**0000000000

(a) The proportion of unique individuals for each generation
(b) The numbers of each module types for each generation
(c) The knob parameters standard deviation for each generation
(d) The distance standard deviation for each generation

Our experiments showed that our approach maintains a high level of diversity. Detailed information about these measures as well as the diversity statistics of our experiments can be found online (Tatar et al., 2015).

5.4 Fitness function

PresetGen measures the computational similarity to a target sound by calculating the Euclidian distance between the audio features of candidate and target sounds, which is a common technique used in automatic preset generation systems (Yee-King, 2011). We focus on perceptual sound similarity in Section 7. *PresetGen*'s multi-objective fitness function uses three objectives: *FFT*, *Envelope* and *STFT* to extract general spectrum, dynamics and spectral envelope, respectively. We used the Euclidian distance between the STFT of candidate (t) and target (c) sound: d_{STFT}. Equation 4 shows the Euclidian distance between the STFT of t and c. *PresetGen* uses the sampling rate of 44,100 Hz, window size N_s of 1024 samples (23 ms), and an overlap of 512 samples (11.5 ms). N_w is the total number of windows.

$$d_{STFT}(t, c) = \sum_{i=1}^{N_w} \sqrt{\sum_{j=1}^{N_s} (t_{i,j} - c_{i,j})^2}. \quad (4)$$

In addition to the *STFT* objective, *PresetGen*'s fitness function includes two more objectives to uncouple the amplitude envelope from the spectral components as much as possible to avoid the premature convergence. Thus, we chose to extract two separate sound features: (1) the FFT (magnitude frequency spectrum) computed on the entire sound without segmentation and (2) the amplitude envelope. Computing the FFT on the entire sound mitigates, to some extent, the effect of the amplitude envelope on the spectrum. We use magnitude spectrum against power spectrum to be consistent with STFT. We extracted the amplitude envelope using the Hilbert transform followed by a low-pass filter. We normalize each sound before the audio feature extraction.

Downloaded by [Simon Fraser University] at 13:21 29 July 2016

Contrary to canonical GA, the NSGA-II uses a selection operator based on non-domination sorting. Using FFT as an objective in the fitness function, an individual with a good set of engine knob parameters would be more likely kept in the population even if it has wrong ADSR knob parameters. Indeed, this individual would have a high fitness value for the FFT and a low fitness value for the Envelope. It would be then kept in the population because it is dominating the population according to the FFT objective. In the canonical GA, this individual would likely be discarded because its fitness value would be affected by a wrong amplitude envelope.

5.5 Selection and stopping criteria

PresetGen uses a population of 500 individuals for each generation. This number of individuals was empirically determined as a good trade-off between performance and computational cost. Moreover, OP-1 is non-deterministic in its design. A parameter set does not necessarily generate the same sound each time. We include the following condition in our system design to overcome the non-determinism of OP-1. The optimization process terminates if the weighted change in the three fitness objectives (given by Equation 5) is less than 10^{-10} over 200 generations. δ_n is the weighted change at generation n, f_k is the best fitness objective score at generation k, N = 200 if $n \geq 200$ otherwise $N = n$. If this condition is never verified, the optimization process stops after 3000 generations. Our experiments indicated that δ_n value of 10^{-10} over 200 generations allows *PresetGen* to converge when we use contrived sounds, and yet reach the maximum number of generations when we use non-contrived sounds that is not in the OP-1's sound space.

$$\delta_n = \sum_{i=1}^{N} \left(\frac{1}{2}\right)^{N-i} (f_{n+1-i} - f_{n-i}). \tag{5}$$

5.6 Pareto front

The cumulative Pareto front is the set of non-dominated individuals for the three objectives explained in Section 2.2. We implement three methods to handle the size of the cumulative Pareto front in *PresetGen*. First, we define a similarity rule that tests whether the chromosome of an individual is already present before adding it to the cumulative Pareto front. This rule cuts the size of the cumulative Pareto front by a factor of more than two. Second, we introduce a similarity rule to limit duplicate individuals in the cumulative Pareto front. We state that two individuals are identical if they have the same engine+LFO+FX types, identical key+octave and the Euclidian distance between the knob parameters is less than 1000 (3% of the knob parameter range). We have determined this value by making tests on large numbers of cumulative Pareto fronts. Depending on the target sound, this similarity rule reduces the size of the cumulative Pareto front to between 10 and 150 individuals while conserving its quality and diversity.

Third, we apply a technique developed by Chaudhari, Dharaskar, and Thakare (2010) to select the most significant individuals in the cumulative Pareto front when the cumulative Pareto front's size is superior to 10 individuals. This approach consists of the following steps:

(1) Apply a k-means clustering algorithm to cluster the solutions enclosed in the cumulative Pareto front. We implement clustering on the OP-1 presets to help the user to identify the presets with different sound engines in the cumulative Pareto front.
(2) Determine the optimal number of clusters, k. The silhouette (Rousseeuw, 1987) of an individual is a measure of how closely it is matched to other individuals within its cluster and how loosely it is matched to individuals of the neighbouring cluster. A silhouette $s(i)$ close to 1 implies that the individual i is in an appropriate cluster, while $s(i)$ close to -1 implies that i is in the wrong cluster. Thus, the average $s(i)$ of the entire cumulative Pareto front is a measure of how appropriately the cumulative Pareto front has been clustered. A value of the average silhouette is obtained for several values of k with $k < 10$. The k that gives the highest average silhouette value is selected. Figure 7(a), Section 6.2.3, is an example of a silhouette plot. Each bar represents the silhouette value, $s(i)$, of an individual in the cumulative Pareto front. This experiment has given the highest average silhouette value with $k = 3$ (Figure 7(a), Section 6.2.3). Y axis represents the cluster number.
(3) For each cluster, select a representative solution. For each cluster, the individual within the cluster that encodes the OP-1 preset that is the closest to the cluster centroid preset is selected as the representative solution.
(4) Analyse the results. At this point, the user can analyse the k representative solutions of the clusters.

5.7 Implementation

In the implementation of *PresetGen*, we use the OP-1's C++ library provided by *Teenage Engineering*. We utilize DEAP framework to implement NSGA-II (Fortin et al., 2012). *PresetGen* uses the Python bindings of the Yet Another Audio Feature Extractor (YAAFE) library for audio feature extraction (Mathieu et al., 2010). We run the Python code on the Bugaboo computing cluster that is part of Westgrid Compute Canada (Westgrid - Compute Canada, n.d.). We use MATLAB (2011) for post-processing with k-means clustering, and webpage generations of experiment results (Tatar et al., 2015).

6. Experiments and results

6.1 Sound collection

Using *contrived sounds* (i.e. sounds generated by the OP-1) as target sounds allows us to validate *PresetGen*'s system design

Downloaded by [Simon Fraser University] at 13:21 29 July 2016

by showing that *PresetGen* can reverse engineer the parameter setting, i.e. the preset. We chose to limit our evaluation to 12 contrived sounds, given the complexity of the algorithm and its running time. We abbreviate these contrived sounds as *conf <number>*. These selected sounds represent a sample of spectrums that was diverse and representative of the OP-1's possible outputs. We focused on having diversity in spectral variation, noisiness and spectral spread (Peeters, 2004). The spectral variation $S_v(t)$ (or spectral flux) represents the amount of variation of the spectrum along time. It is computed from the normalized cross-correlation between two successive amplitude spectrums $a(t-1)$ and $a(t)$. We use a window size of 1024 samples and a hop size of 512 samples with 44,100 Hz sampling rate to calculate the spectral variation.

$$S_v(t) = \frac{\sum_k (a_k(t) - a_k(t-1))^2}{\sqrt{\sum_k a_k(t-1)^2}\sqrt{\sum_k a_k(t)^2}}, \qquad S_v = \sum_t S_v(t). \tag{6}$$

$a_k(t)$ is the kth bin of the amplitude spectrum at time t. The set of non-contrived sounds consists of both the sounds with a stationary spectrum and the sounds with a dynamic spectrum, as measured by their respective spectral variation. This measure does not include perceptual facts. We utilize spectral variation to study *PresetGen*'s fitness function performance on target sounds with different spectral properties. We used each engine, LFO and FX at least once to generate this set of contrived sounds (Table 3). For each of these sounds, we also give its overall spectral variation S_v. We distinguish two groups: the sounds with a stationary spectrum ($S_v < 10^{-5}$) and the sounds with a dynamic spectrum ($S_v \geq 10^{-5}$).

A second evaluation was performed on 12 non-contrived sounds, including synthetic sounds, instrument sounds, a male voice and a cat sound. These sounds were also chosen to have a diversity in spectral variation, noisiness and spectral spread. As with the contrived sound, we distinguish two groups: the sounds with a stationary spectrum ($S_v < 10^{-5}$) and the sounds with a dynamic spectrum ($S_v \geq 10^{-5}$). Both contrived and non-contrived sounds are available for listening online (Tatar et al., 2015).

6.2 The detailed statistics of two sounds

In this section, we present the detailed statistics of two runs with two target sounds, *conf4* and *cat*, exemplifying contrived and non-contrived sounds respectively. Section 6.3 presents the global statistics of multiple runs with all target sounds. The statistics and results for all runs are available online (Tatar et al., 2015).

6.2.1 Fitness objectives

The goal of the GA is to minimize the three objectives (Envelope Euclidian distance, FFT Euclidian distance and STFT Euclidian distance). Figure 4 shows the reduction of the FFT distance over the generations. On both graphs, a viewer can observe plateaus, also called *punctuated equilibriums*, that can be interpreted as periodic improvements in fitness. One can also distinguish the *exploration* phase and *exploitation* phase on these graphs.

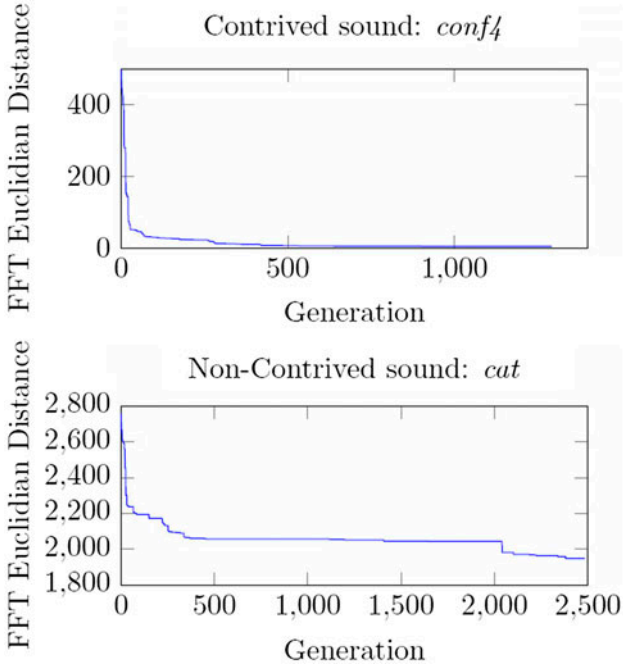

Fig. 4. Fitness objective: the minimum FFT Euclidian distance (in the population) to the target sound.

The *exploration* phase happens at the early stage of the evolution with a quick and significant improvement in the fitness. The *exploitation* phase follows with episodic and small improvements in the fitness. Exploration also continues during exploitation. The length of these phases can vary depending on the nature of the target sound (Lobo, Lima, & Michalewicz, 2007). For example, a viewer can see that the *exploration* phase lasts for 50 generations in the contrived sound case and around 350 generations in the non-contrived sound case. Moreover, the range of the final objective fitnesses differ for the contrived sound trials and for the non-contrived sound trials. For example, the final FFT fitnesses vary between 0 and 500 for the contrived sound trials and between 2000 and 3000 for the non-contrived sounds trials. This difference illustrates the fact that, as expected, it is more difficult to approximate non-contrived sounds (which have large final fitness values) than contrived sounds (which have smaller final fitness values).

6.2.2 Distance/fitness correlation

Regarding the contrived sounds, another way of tracking the improvements of the system for the non-contrived sounds is to look at the parameter distance to the target parameters over the generations (Jones & Forrest, 1995). We considered two distances: the Euclidian distance and the Hamming distance.

Table 3. Parameters of the contrived sounds.

Conf. Id	Engine	FX	LFO	key	octave	S_v	Stationary
conf0	FM	Grid	No	17	4	1.252×10^{-4}	N
conf1	Digital	Delay	No	1	−1	7.641×10^{-6}	Y
conf2	Cluster	Delay	Value	16	3	6.625×10^{-4}	N
conf3	Digital	Punch	Tremolo	16	3	1.172×10^{-4}	N
conf4	Digital	Delay	No	9	2	9.301×10^{-6}	Y
conf5	FM	No	No	0	2	3.055×10^{-4}	N
conf6	FM	No	No	11	1	7.128×10^{-5}	Y
conf7	String	No	No	20	−3	3.055×10^{-4}	N
conf8	Cluster	No	No	12	0	6.740×10^{-6}	Y
conf9	Pulse	No	No	9	−1	2.279×10^{-4}	Y
conf10	Phase	No	No	9	−4	1.154×10^{-4}	N
conf11	Phase	Punch	Element	16	1	9.4187×10^{-6}	Y

Downloaded by [Simon Fraser University] at 13:21 29 July 2016

We define the Euclidian distance between two individuals in Equation 7.

$$d_e(r, s) = \sqrt{\sum_{i=1}^{N}(r(i) - s(i))^2}, \tag{7}$$

where r and s are two individual sets of parameters, and N is the total number of parameters. $r(i)$ (resp. $s(i)$) are the ith parameter of individual r (resp. $s(i)$).

The Hamming distance between the two bit strings of chromosomes is defined in Equation 8.

$$d_h(u, v) = \frac{c_{01} + c_{10}}{n}, \tag{8}$$

where c_{ij} is the number of occurrences of $u[k] = i$ and $v[k] = j$ for $k < n$, n being the number of bits. We study the Hamming distance as well because *PresetGen* uses bitwise representations of OP-1 parameters to find a contrived sound target's preset.

Figure 5 shows the minimum of these distances to the target individual/chromosome over the generations for the *conf4* sound. Contrary to the fitness objectives (Figure 4), the curves are not monotonic decreasing. The Euclidian distance even appears to increase over the generations. It is expected that Euclidian distance varies in a non-monotonic, non-predictable way. Moreover, it is possible to achieve a good approximation of the target sound using completely different parameters than the ones used initially to produce it. For this reason, tracking Euclidian distance is of limited relevance to evaluate the performance of *PresetGen*. However, observing the correlation between fitness and distances presents the difficulty of our problem.

We study the correlation of the distance—between the target preset genotype and an individual's genotype—and the fitness value to evaluate the difficulty of the problem being solved. We plot the average FFT distance to the target sound per generations against the average Euclidian and Hamming distances of phenotypes (OP-1 presets) to the target preset (Figure 6). One can see that, on average, the Euclidian and Hamming distances

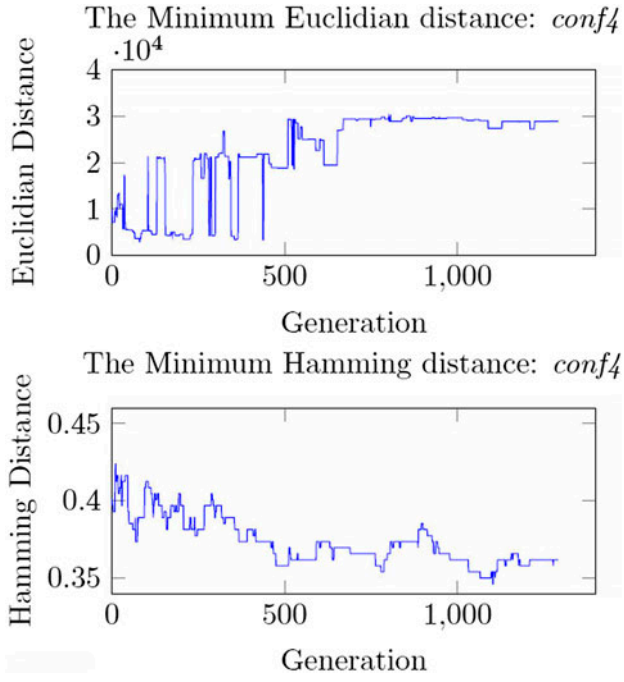

Fig. 5. The minimum distance (in the population) between the target preset's genotype and the individual's genotype per generation.

are well correlated to the fitness with a correlation coefficient equal to 0.67 for the Euclidian distance, and 0.70 for the Hamming distance. These results show that, on average, the fitness leads the GA to the region in the parameter space where the target is located. We can also observe, with the cluster of points around the coordinates $[3 \times 10^4, 0]$ of the Euclidean distance correlation, that a single large optimal plateau exists.

6.2.3 *Pareto front analysis and clustering*

Trials show that, most of the time, the cumulative Pareto front is too big to be easily handled by the final user. The idea is to give a sample of individuals (less than 10) that is

Downloaded by [Simon Fraser University] at 13:21 29 July 2016

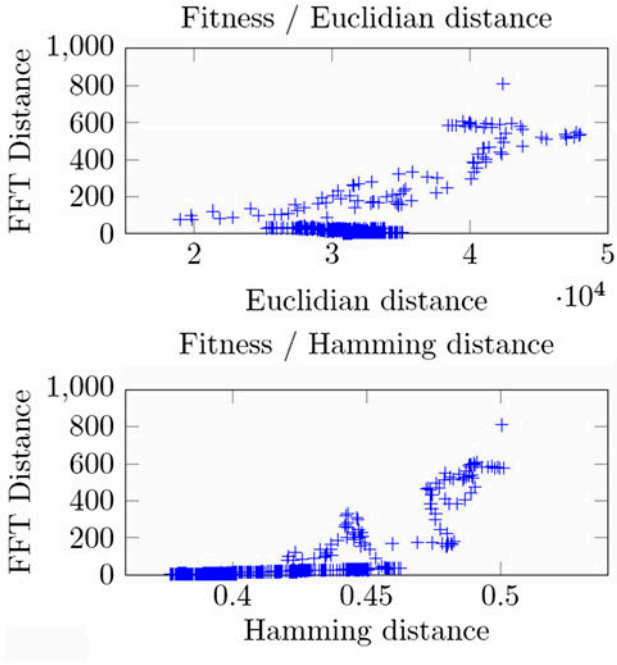

Fig. 6. Average fitness versus distance, each sample represents one generation.

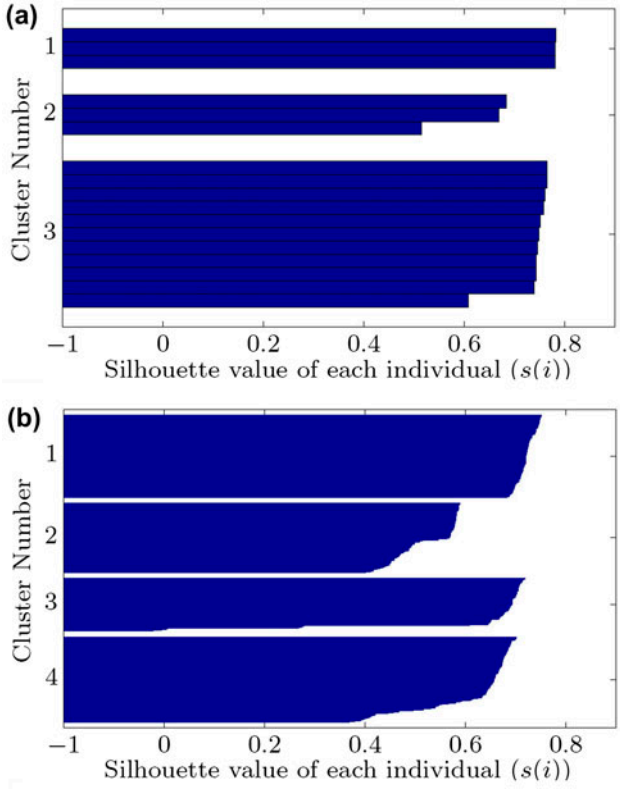

Fig. 7. Cumulative Pareto front individuals' silhouette values: (a) contrived sound—*conf4*; (b) non-contrived sound—*Cat*.

a good representation of diversity in the cumulative Pareto front. As described in Section 5.6, we apply a k-means clustering algorithm to cluster on the solutions enclosed in the cumulative Pareto front. As the clustering is done on the OP-1 knob parameters, the user can easily retrieve all the individuals enclosed in the given cluster, starting with the centroid individual. Indeed, we assume that going from the centroid individual to any individuals in the cluster, the user would slightly modify the centroid individual's parameters. Moreover, we also assume that individuals in the same cluster sound similar, as their OP-1 configurations are similar. The k-means clustering algorithm requires that the number of clusters k to be chosen before running the algorithm. We use the silhouette method (described in Section 5.6) to determine the number of clusters. Figure 7(a) and (b) show the silhouette values of cumulative Pareto front individuals for the experiments with the *conf4* and *cat* target sounds, respectively. The experiment with *conf4* target sound has three clusters in the final cumulative Pareto front, whereas the experiment with *cat* target sound has four. Each bar represents the silhouette value of an individual. One can observe these bars more easily on Figure 7(a), as the cumulative Pareto front is small for the *conf4* target sound. As one can see on both Figure 7(a) and (b), the average silhouette is close to 1 and there is no individual with a small or negative silhouette.

Figure 8(a) and (b) give a 3D representation of the cumulative Pareto front over the three objectives (FFT, STFT and Envelope). As described in the previous section, the clustering is done on the OP-1 parameters and not on the fitness objectives. However, a viewer can observe that the cumulative Pareto front is also well clustered over the three objectives. It makes sense because neighbours in the OP-1 parameter space should also be neighbours in the objective space. However, there are outliers (for example, one individual belonging to the cluster 3 on Figure 8(a)). It could be because the clustering is not perfect, but it can also illustrate the inverse correlation between parameter distance and fitness objectives (see Section 6.2.2).

The individual in each cluster that is the closest to the centroid is chosen to represent its clusters. Figures 9(a) and 10(a) both show a comparison between the centroid individual spectrogram and the related target sound spectrogram for the *conf4* and *cat* sounds, respectively. For the contrived sound *conf4*, the two spectrograms look similar. However, the fitness values for the FFT and STFT on Figure 8(a) are small but not equal to zero. Hence, the match is not perfect, but it is very close, as it is not possible to perceive the difference when we listen to the two sounds. For the non-contrived *cat*, the two spectrograms do not look as similar as the contrived sound case. With a non-contrived sound, it is not possible to know in advance if we can generate the exact match of a non-contrived target sound with the OP-1 synthesizer. However, we see strong similarities in the frequency range in both spectrograms and also in the spectral envelope. Figure 9(b) (respectively Figure 10(b)) shows a comparison between the centroid individual waveform and the related *conf4* (respectively *cat*) target sound waveform. For both contrived and non-contrived sounds, we can see that the amplitude envelopes are either close (*cat*) or almost identical (*conf4*).

Downloaded by [Simon Fraser University] at 13:21 29 July 2016

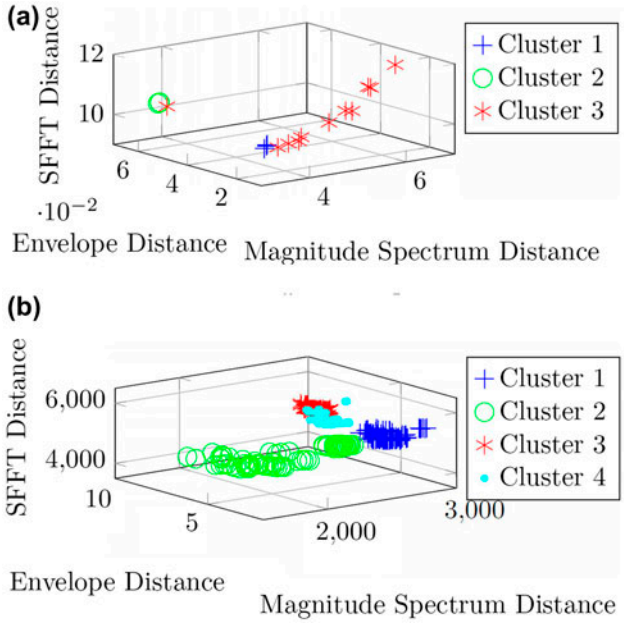

Fig. 8. Visualization of clusters in the cumulative Pareto fronts: (a) contrived sound—*conf4*; (b) non-contrived sound—*Cat*.

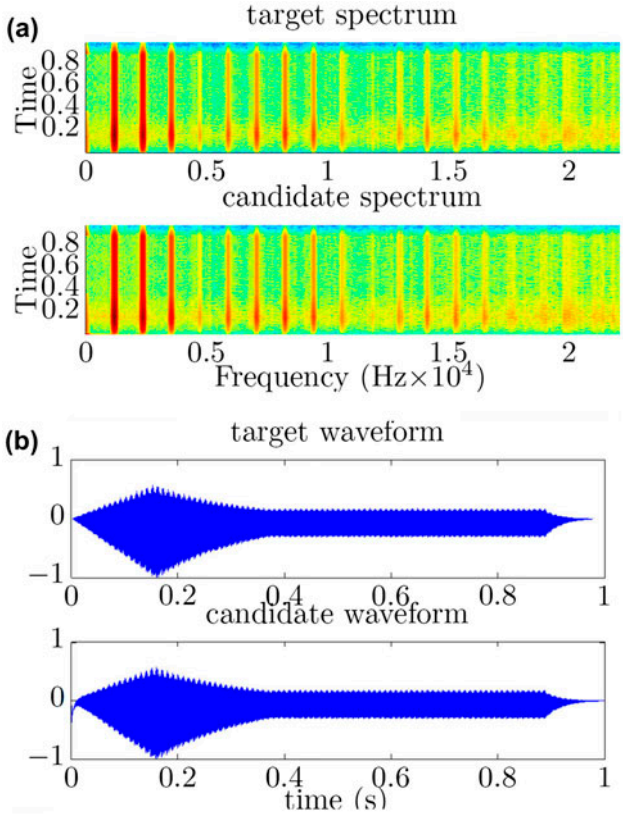

Fig. 9. Contrived sound (*conf4*): centroid individual for the Cluster 1. (a) Spectrograms and (b) waveforms.

6.3 Statistics of multiple runs

6.3.1 Bootstrapping

We ran *PresetGen* 10 times for each target sound. We used Bootstrapping to obtain estimates of summary statistics

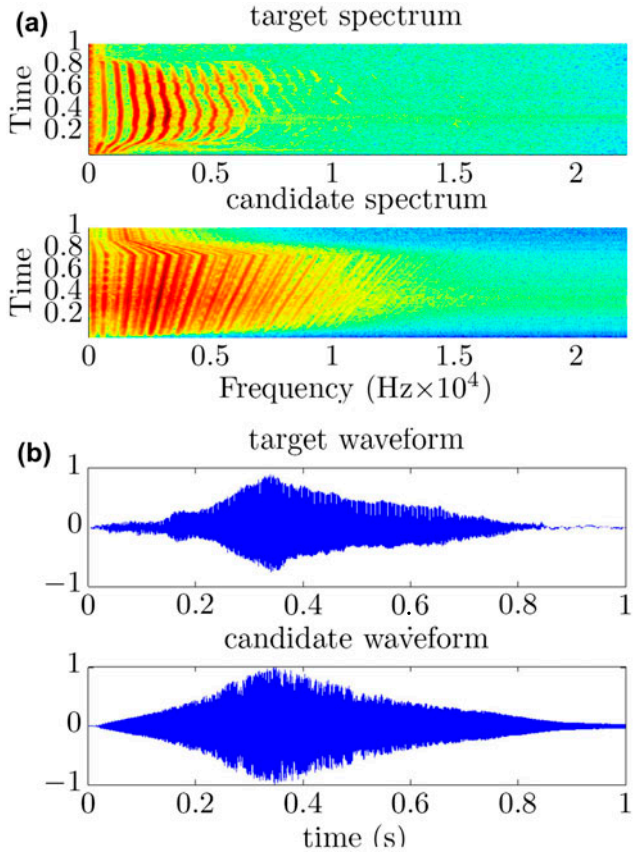

Fig. 10. Non-contrived sound (*cat*): centroid individual for the Cluster 1. (a) Spectrograms and (b) waveforms.

(Johnson, 2002). The bootstrapping procedure repeats and excludes data from the original data set to create a bootstrap sample. A bootstrap sample has the same amount of data as the original data set. However, it is not identical to the original data set. We repeat the generation of a bootstrap sample a large number of times (1000 in our case). For each of these bootstrap samples, we compute the desired statistic that provides an estimate of the distribution of the descriptive statistics. This technique makes the information extraction possible when the sampling size is small, as in the case of *PresetGen* due to the time complexity of the problem.

For each of the measures described above, we used bootstrapping to get an estimate of its minimum, maximum, mean and standard deviation. We also used the bootstrap shift method test (Johnson, 2002) to assess the significance of every comparison we performed. This test has the advantage of being distribution-free and of scaling well with small sample sizes.

6.3.2 Measurements of the solution quality

We evaluated the solution quality and the running time in the descriptive statistics of PresetGen experiments. The solution quality was measured differently for the contrived sounds and the non-contrived sounds. Regarding the contrived sounds, we already know what the target OP-1 presets are. However,

Downloaded by [Simon Fraser University] at 13:21 29 July 2016

OP-1 is non-deterministic. An OP-1 preset does not generate the exact same sound each time. Therefore, we generate 10 sounds using a target preset in addition to the target sound, and compare them to the target sound. With a determinist synthesizer, their fitness objective values (FFT, envelope and STFT) would be equal to zero, but this is not the case with the OP-1. For each objective, we define the best possible fitness value F_b as the minima of the fitness value over these 10 sounds. For each objective, we calculate the error Δ_r for each run, subtracting this best possible fitness value F_b to the best fitness objective value obtained in the particular run f_r (see Equation 9).

$$\Delta_r = f_r - F_b. \tag{9}$$

Regarding the non-contrived sounds, we are only able to measure the final fitness values for the three objectives at the end of the evolution. In both cases, the running times are measured by the number of generations before the GA reaches the stopping criteria (*nbGen*). Sections 6.3.3 and 6.3.4 present the solution quality measurements of experiments with contrived and non-contrived sound, respectively.

6.3.3 Contrived sounds

Figure 11 shows the number of generations before reaching the stopping criteria for each target configuration. Table 4 describes the statistics of the proportion of module types in the population over the generations. *Prop. choice* is the proportion of runs in which one type was taking over in the population. *Take over gen* is the generation from which one type was taking over. *Accuracy* is the proportion of runs choosing the correct type when one type was taking over in the population. The OP-1 has 24 keys on its keyboard, and it is possible to change the octave from −4 to 4. There is an overlap of 12 keys between two consecutive values of an octave. It is possible to produce the same note using two different combinations of octaves and keys. The last line of Table 4 takes into account this particularity and considers the selected note, rather than the octave and the key taken separately.

The results show that *PresetGen* performs well at finding the right engine type (90% prop. choice; 80% accurate) and the right note (74% prop. choice; 69% accuracy). However, we have lower accuracy for the LFO (43 prop. choice; 22% accuracy) and FX type (42% prop. choice; 18% accuracy) than the engine type. A possible interpretation of these results is that the engine type and the note have a greater influence on the output sound than the LFO or FX type. The LFO and FX type do not change the nature of the output sound, but only alter it. Thus, it is more challenging to determine the right type for the FX and LFO.

The correlation between the Euclidian/Hamming distances of individuals to the optimal solution and the fitness objectives are a good indicator of the problem difficulty (Jones & Forrest, 1995). We compute the mean and standard deviation of these distances and the three fitness objectives for each generation. We calculate a *global correlation coefficient* between the average Euclidian/Hamming distances of individuals to the optimal solution and average of each fitness objectives. Additionally, we compute a *local correlation coefficient* between the average Euclidian/Hamming distances of individuals to the optimal solution and the average of each fitness objectives. Table 5 shows these correlation coefficients. The global correlation coefficients are very high for the Euclidian and Hamming distances. These results explain the tendency of the GA to converge quickly to a punctuated equilibrium of low fitness for the three objectives. This tendency is also an illustration of *PresetGen*'s ability to converge toward sounds perceptually similar to the target sound. The low local correlation coefficients show that, once a promising location of the fitness landscape was identified, it is challenging to fine tune the input parameters to converge exactly to the target parameters. One explanation is that the non-deterministic nature of the synthesizer (which causes noise in the evaluation) makes perfect tuning impossible. Some knobs also have different

Table 4. Statistics about modules types. C: Contrived sounds, NC: Noncontrived sounds.

	Prop. choice		Take over gen		Accuracy
	C	NC	C	NC	C
Engine	0.90	0.74	139	129	0.80
FX	0.42	0.44	322	270	0.18
LFO	0.43	0.45	240	317	0.22
Key	0.77	0.38	122	265	0.52
Octave	0.91	0.57	109	174	0.62
Note	0.74	0.38	129	273	0.69

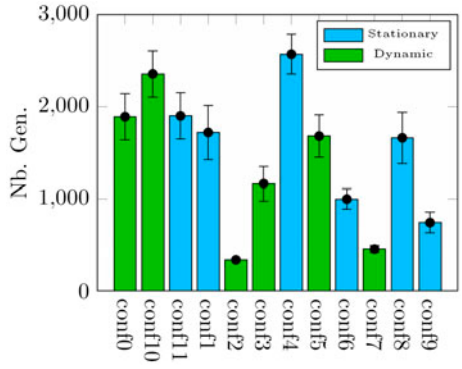

Fig. 11. Number of generations before reaching the stopping criteria: contrived sounds.

Table 5. Mean and SD for the correlation coefficients.

		FFT	Env	STFT
		Mean(SD)	Mean(SD)	Mean(SD)
Euclidian	**Global**	0.35(0.46)	0.37(0.45)	0.34(0.41)
	Local	0.04(0.40)	0.09(0.43)	0.09(0.45)
Hamming	**Global**	0.54(0.34)	0.57(0.27)	0.59(0.31)
	Local	0.04(0.4)	0.006(0.42)	0.06(0.38)

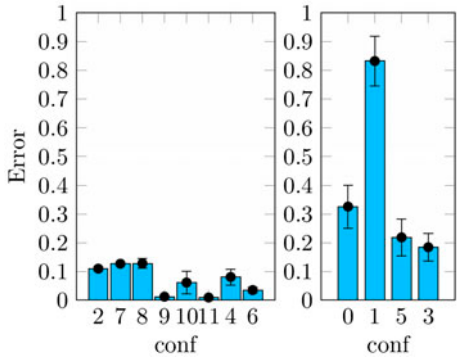

Fig. 12. Error: Envelope—contrived sounds.

sensitivity; therefore, a change in one knob can either cause a large change in the output or no change at all.

We refer to module combination as an OP-1 preset without the knob parameters. For example, {FM synthesis engine, FX delay, LFO element, key 22} is an example of a module combination. The number of distinct module combinations was very low in the cumulative Pareto front ($\mu = 3.0$, $SD = 0.2$ over 10,080 possible combinations). These findings suggest that the GA successfully identified a limited number of promising locations in the parameter space that dominate all others. We also notice that, as wanted, each cluster contains only one Engine/LFO/FX combination. A cumulative Pareto front provides flexibility to the user with a set of candidate sounds rather than a single sound. The user can make the final choice. Regarding non-contrived sounds, the user can decide what type of perceptual error is more acceptable as interesting.

Figure 12 shows that *PresetGen* successfully approximates the amplitude envelope of the target sound as shown by the very low errors for the envelope objective (μ = 0.20, SD = 0.02). Figure 13 shows the error over the best possible FFT and STFT objective values. As measured by their respective spectral flux, *conf0, conf2, conf3, conf5, conf7* and *conf10* are the configurations generating dynamic spectra. The other configurations generate stationary spectra. We see that the performances of the GA are not significantly better for the target sounds with stationary spectra ($p = 0.07$) than for the target sounds with dynamic spectra ($p = 0.08$). However, we can still observe differences in the GA performances for different groups of target sounds. The group with *conf9* and *conf10* has the lowest FFT and STFT errors. *conf10* presents a STFT under the form of a sawtooth wave over time. This common shape doesn't seem to be difficult to approximate for our system, even if the spectrum is dynamic. The group with *conf0, conf1, conf3* and *conf5* contains mostly target sounds with dynamic spectra with the exception of *conf1*. The error for this group is the highest. The spectral energy of *conf1* is mainly concentrated in a narrow frequency band. Our system seems to fall into a local minima for a non-periodically time-changing spectrum. These results show that the nature of a target sound has an impact on the final fitness objective errors of *PresetGen*.

6.3.4 Non-contrived sounds

Using non-contrived sounds, we present the distance-fitness corelation to prove that *PresetGen* converges to target sounds (see Section 6.2). Regarding non-contrived sounds, the structure and complexity of the fitness landscape depend largely on the chosen target sound. OP-1's sound space does not cover the whole digital sound space. Even if *PresetGen* finds the optimal solution in the synthesizer's search space, we can expect a high error if a non-contrived target sound is not actually reproducible with the OP-1.

The mean of *nbGen*—the number of generations before reaching the stopping criteria (see Section 5.5)—is significantly larger for the contrived sounds ($\mu = 1431$, $SD = 86$) than for the non-contrived sounds ($\mu = 1070$, $SD = 50$); $p < 0.001$. These results may seem surprising, as the resynthesis problem for a non-contrived sound is more complex than for a contrived sound. However, we design our algorithm to stop if the cumulative fitness improvement is almost null for each objective (see Section 5.5). *PresetGen* reaches a level of fitness (where improvements are more difficult to obtain) faster for the non-contrived sounds than for the contrived sounds, because the potential improvements are more limited in the non-contrived case than in the contrived case. Moreover, observing Figure 4, we can clearly see that contrived sound, *conf4*, reaches the lower fitness value faster than the non-contrived sound, *cat*, at the beginning of the algorithm. The convergence is faster but with a long tail.

Table 4 describes statistics of the proportion of module types in the population through the generations. As with contrived sounds, a single engine was quickly taking over. However, contrary to the trials with contrived sounds, no key was clearly taking over (Prop. choice 38% versus 77% for contrived sounds) because most of the non-contrived sounds do not have a clearly identified stationary fundamental frequency (*cat*, DX-7 and Moog synthesizer sounds with pitch modulation). FX and LFO types were, as with contrived sounds, still challenging to set for the GA (FX prop. choice 44%, LFO prop choice 45%).

Figures 14 and 15, respectively, show the errors for the FFT/STFT and the Envelope. First, we can hear that the cumulative Pareto front sounds are perceptually similar to the targets, which is of importance for real world applications. Regarding the number of type combinations, the cumulative Pareto fronts of non-contrived sounds are also significantly more diverse (μ = 4.3, SD = 0.3) than the ones we got using contrived target sounds (μ = 3.0, SD = 0.2); $p < 0.001$. The cumulative Pareto fronts of the non-contrived sounds are also significantly more populated (μ = 306, SD = 11; μ = 83, SD = 21); $p < 0.001$. These differences are caused by the larger problem complexity with the non-contrived sounds. With the concept of clustered cumulative Pareto front, the user receives a set of OP-1 presets that produces sounds perceptually similar to the target sound. These OP-1 presets do not automatically involve the use of the same engine, LFO or FX, which gives users several alternatives of variable quality to approximate a

Downloaded by [Simon Fraser University] at 13:21 29 July 2016

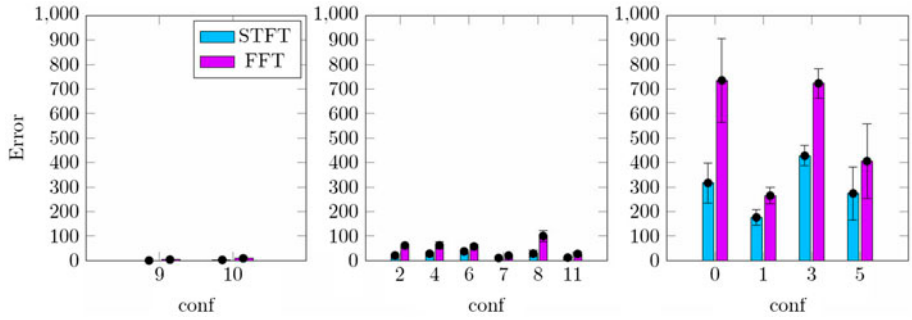

Fig. 13. Error: FFT and STFT—contrived sounds.

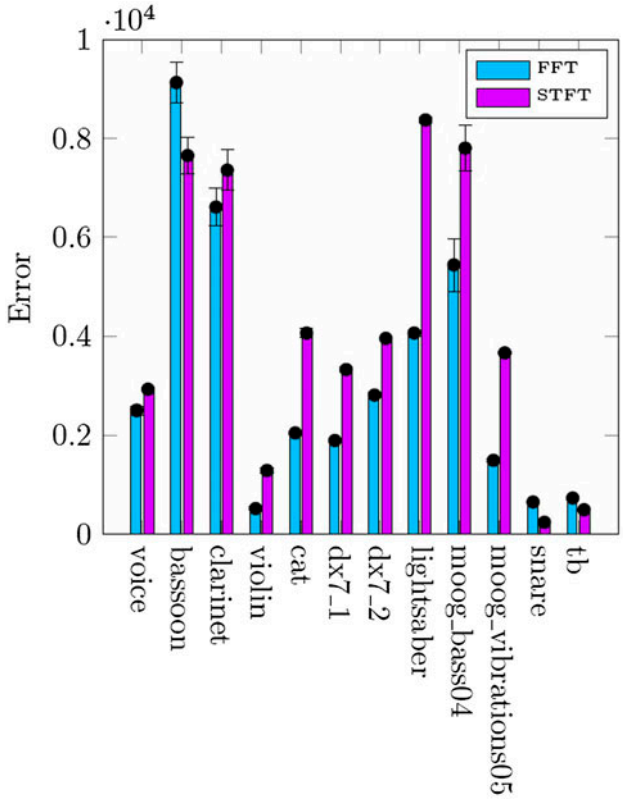

Fig. 14. Fitness objectives: FFT and STFT—non-contrived sounds.

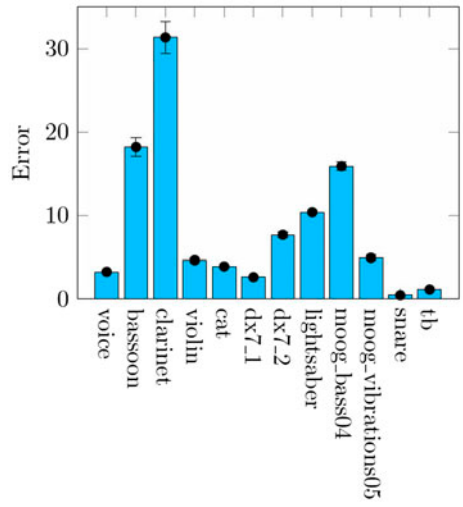

Fig. 15. Fitness objectives: Envelope—non-contrived sounds.

Table 6. Best fitness objective values. C: Contrived sounds, NC: Noncontrived sounds.

	FFT		Env.		STFT	
	Mean	**SD**	**Mean**	**SD**	**Mean**	**SD**
C	130.34	17.15	0.20	0.02	248.92	31.24
NC	3163.5	242.67	8.66	0.78	4299.5	262.62

given target sound. The best fitness values for the three objectives, shown in Table 6, are significantly worse for the non-contrived sounds than for the contrived sounds as expected ($p < 0.001$, $p < 0.001$, $p < 0.001$).

7. Empirical evaluation with non-contrived sounds

PresetGen is meant to be used by humans. Hence, comparing its performance with a human sound designer and looking at the perceived quality of *PresetGen*'s results is of interest. There are only two automatic preset generation studies with an empirical evaluation experiment. Mitchell (2010) conducted an empirical evaluation of his FM Modulation Audio Synthesis Parameter Optimization. Mitchell's evaluation consisted of two listening tests. In the first listening test, Mitchell conducted an experiment to research the correlation between fitness error ranking and human perception ranking, thus addressing the—still open—problem of sound similarity. In the second listening test, Mitchell conducted a qualitative research with five musical instruments sounds. Participants commented on matching sounds in terms of pitch, amplitude envelope and timbre.

Yee-King and Roth (2008) also evaluated their system, *Synthbot* with expert human users. The application included two VSTi synthesizers: the mdaDX10, a single module synthesizer with sixteen parameters, and the mdaJX10, a substrative

Downloaded by [Simon Fraser University] at 13:21 29 July 2016

Downloaded by [Simon Fraser University] at 13:21 29 July 2016

synthesizer with one noise oscillator and 40 parameters. In the first part of the experiment, 10 expert users implemented two sounds: one real sound and one synthesized sound for each of these two synthesizers. They reported the participants' comments on each matching sound.

We conducted an empirical evaluation experiment to measure *PresetGen*'s performance on perceptual sound similarity. We compare the performance of three (human) sound designers and *PresetGen* on the task of OP-1 preset generation with non-contrived target sounds. We compared the preset generation performances of designers and *PresetGen* using *PresetGen*'s fitness function. *PresetGen* converges to more than one preset for a given target sound. For simplicity, we choose only one *PresetGen*'s candidate sound to be compared with sound designers' candidate sounds for each target sound.

In this experiment, we first compared *PresetGen* with one human sound designer on the task of OP-1 preset generation for non-contrived sounds. We researched the relationship between *PresetGen*'s fitness function results and human listener evaluation. Experiment participants evaluated the similarity between the target sounds and matching sounds designed by the (human) designer and *PresetGen*. Second, we expanded our experiment to three sound designers to generalize that *PresetGen* is human-competitive.

We limited the number of target sounds to eight. The aim was to have an experiment that lasted no longer than 15 min of listening evaluation. We covered diversity on three objectives of *PresetGen*—STFT, Envelope and FFT—by selecting one sample of each of the following acoustic sounds; *ak47, chicken clucking, bass guitar, trumpet, gyil,*[3] *construction noise, knife sharpener* and *frog* (Tatar et al., 2015). These were restricted to have a stationary fundamental frequency and an applicable envelope considering the capabilities of OP-1.

7.1 Method of the experiment

7.1.1 Design

We used repeated-measure design, in which every participant was exposed to all conditions. The dependent variable is *participant similarity rating*, whereas independent variables are

(1) *sound designer* with two-factor levels (*human sound designer* and *PresetGen*)
(2) *target sounds* (eight different sound samples)
(3) *sound attributes* (*general, pitch, envelope and timbre*)

We also conducted four paired t-test analyses for each sound attributes of each target sound, 32 paired t-tests in total. In these tests, the dependent variable is *participant similarity rating*, whereas the independent variable is a category including two factor levels of *the designer* and *PresetGen*.

7.1.2 Participants

Fourteen auditors (as opposed to sound designers) were recruited from the *IAT 340 Sound Design* class in the School of Interactive Arts and Technology at Simon Fraser University so that they had basic experience in (and introductory knowledge of) sound design, sound synthesis and related terms such as timbre, pitch and envelope.

One participant didn't provide the demographics and answers to experience-related questions. Of the remaining 13 participants, eight were female and five were male. 71% of participants had less than 6 months of sound design experience, whereas 21% of participants had 6 months to 1 year of experience in sound design. 29% of participants had no musicianship experience, whereas 21% of participants had less than 6 months of musicianship experience. 14% of participants had 1 to 3 years of musicianship experience. 7% of participants had 5 to 10 years of musicianship experience while 14% of participants had 3 to 5 years of musicianship experience. 7% of participants had more than 10 years of musicianship experience.

A (human) sound designer, i.e. user, tried to generate the same target sounds with the OP-1 and the candidate sounds were recorded. The designer was a 26-year-old male with 5 years of sound design experience and 10 years of musicianship experience. He used the OP-1 for a month before the experiment.

7.1.3 Procedure

The experiment took 10 min at most. It was carried out in one session, at the beginning of the course's weekly workshop. Participants used full range circumaural monitoring headphones and the same type of computers with *M-Audio Fast Track Pro* interfaces. The experiment was implemented with an online survey created with *fluidsurveys*.[4] Hence, a browser, mouse and keyboard were used as the user interface. The actual interface is pictured in Figure 16.

PresetGen generated the parameters that could approximate each target sound with OP-1. We present the detailed analysis of these runs online (Tatar et al., 2015). The design of the experiment involved participants evaluating one target sound and one matching sound at a time because we needed participants to evaluate how similar a matching sound is to its target sound. For each target sound, there were two matching sounds to be evaluated, one generated by the designer and one generated by *PresetGen*. The presentation order was random and balanced within the participant population. Participants could listen to the sounds as much as they wanted. We did not provide any information on how the sounds were generated, to prevent any bias against computational creativity (Moffat & Kelly, 2006). Participants answered four questions for each matching sound. The questions asked about general similarity, similarity in terms of pitch, similarity in terms

[3] A type of Xylophone from Ocenia.

[4] Phttp://fluidsurveys.com/s/syntheval/

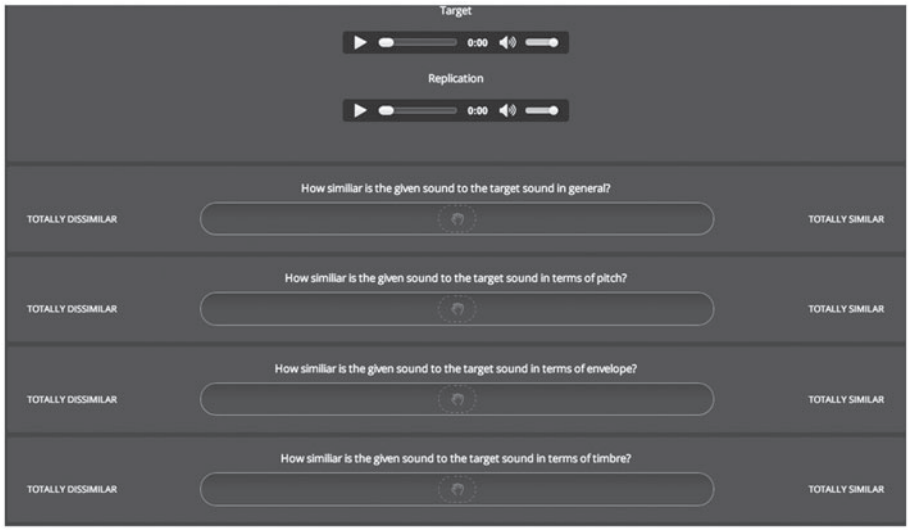

Fig. 16. User interface used to evaluate each matching sound on different sound attributes in terms of similarity to the target sound.

Downloaded by [Simon Fraser University] at 13:21 29 July 2016

of timbre and similarity in terms of envelope; respectively. Participants answered these questions based on a 100-point scale in which 100 is the most perceptually similar and 0 is the most perceptually dissimilar. We used a similarity scale instead of ranking the candidate sounds with their similarity to the target sound to overcome the disadvantages related to the ordinal evaluation. First, the ordinal evaluation cannot show if two candidate sounds have an insignificant difference in terms of similarity to the target sound. Participants have to decide that one candidate sound is better than others. Second, ordinal similarity evaluation cannot provide a metric of similarity. We can not compare the similarity measure of candidate sounds that are generated to match different target sounds. However, analysis of a cardinal evaluation provides a comparison between matching sounds of different target sounds.

7.2 Results

7.2.1 Quality comparison

We compared the average participant similarity ratings of *PresetGen*'s and the designer's matching sounds in terms of general similarity, pitch, envelope, and timbre. We conducted a repeated-measures ANOVA[5] with three independent variables—*audio attributes, user/PresetGen* and *target sounds*—and one dependent variable—*similarity to the target sound*. We analysed tests of within-participant effects for each independent variable and combinations of two independent variables.

The analysis shows that participants rated *PresetGen*'s matching sounds as more similar to the target sound than the (human) designer ones with a mean difference of 16.64 ($F(1, 13) = 48.54$, $p < 0.001$, $w^2 = 0.789$).[6] Analysing sound attribute category tests within-participant effects, participants rated *PresetGen*'s matching sounds significantly higher than the designer's matching sounds for all sound attribute categories ($F(3, 39) = 3.62$, $p = 0.021$, $w^2 = 0.218$) as shown in Figure 17. Additionally, for this independent variable, the Mauchly's test of sphericity[7] showed that sphericity has not been violated ($p = 0.630$). The error bars presented in both Figures 17 and 18 show that the standard deviation from the mean is small. Hence, the participants agreed on a similarity rating for all matching sounds.

We have expanded the analysis by examining each sound. Paired t-tests (mentioned in Section 7.1.1) shows that the difference in the average similarity rating between *PresetGen* and the designer is insignificant for matching sounds of target sounds; *gyil*, *ak47* and *bass*, for all sound attribute categories. The common property of these sounds is that they have an impulse or a short burst, and they have percussive characteristics. Figure 18 illustrates the average similarity ratings of each matching sound. Paired t-tests' significance values are illustrated online (Tatar et al., 2015).

7.2.2 The time complexity

Although *PresetGen*'s matching sounds are rated more similar in quality, the designer was faster than *PresetGen*. The designer could match three sounds in an hour with OP-1, whereas *PresetGen* generated parameters to match a 2 s target sound in 5 h with an evolving population of 500 individuals over 1000 of generations on the Westgrid's Bugaboo computing cluster with 50 cores (Westgrid - Compute Canada, n.d.). We

[5]Analysis of Variances

[6]w^2 is the effect size and represents what percentage of the variance on the data can be explained by the independent variable variance.

[7]Sphericity is defined as 'the homogenity of the covariance between pairs of conditions' (Hinton, 2004).

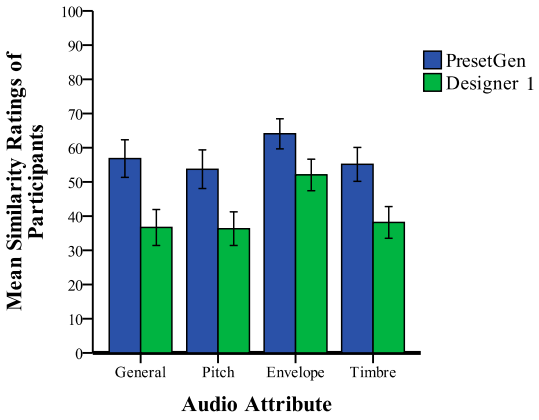

Fig. 17. *PresetGen*'s and the designer's matching sounds' average similarity ratings by audio attributes, error bars represent 95% confidence interval.

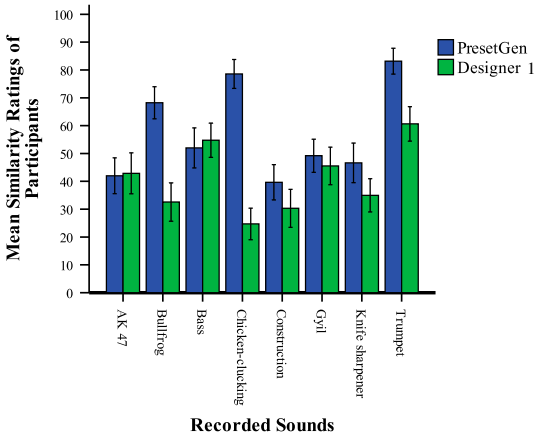

Fig. 18. *PresetGen*'s and the designer's average similarity ratings for each target sound, error bars represent 95% confidence interval.

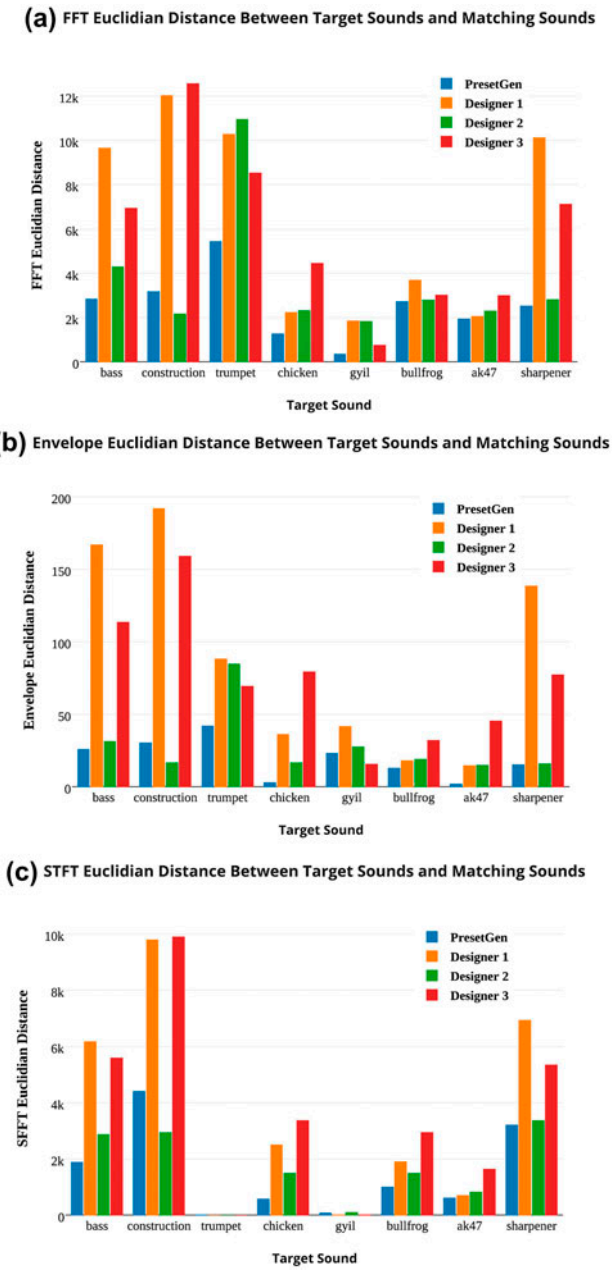

Fig. 19. Fitness function objectives comparison of three sound designers and *PresetGen*. (a) FFT Euclidian distance between matching sounds and target sounds. (b) Envelope Euclidian distance between matching sounds and target sounds. (c) STFT Euclidian distance between matching sounds and target sounds.

presented details of our implementation in Section 5.7. It is also important to note that in the case of a sound designer, the designer chooses to stop, whereas *PresetGen* stops if one of the stopping criteria is achieved. That being said, *PresetGen* can be improved with optimization and advancements in the computational processing speed. We explain our future work in Section 8.

7.3 Expansion of empirical evaluation experiment to multiple sound designers

One can argue that empirical evaluation with non-contrived sounds included only one (human) sound designer and results were dependent on the designer's capabilities. Therefore, we expanded our empirical evaluation experiment with a fitness value comparison of three (human) designers' and *PresetGen*'s matching sounds. *Designer 1* is already mentioned in Section 7.1.2. We asked two more sound designers to match the same eight target sounds that we used in our empirical evaluation experiment, using OP-1. We compared the fitness values of matching sounds of designers and *PresetGen* using *PresetGen*'s fitness function. Figure 19 shows comparisons of fitness values in the three objectives that we used in our multi-objective implementation; *FFT*, *Envelope* and *STFT*. Figures 19(a) to (c) show that *Designer 2* did slightly better than *PresetGen* to match the target sound *construction* in all three objectives. Also, *Designer 3* matched the target sound *gyil* better than *PresetGen* in *Envelope* and *FFT* fitness objectives. Other than these exceptions, *PresetGen*'s matching sounds gave lower fitness values than (human) sound designers' did. Therefore, *PresetGen*'s matching sounds were closer to the

Downloaded by [Simon Fraser University] at 13:21 29 July 2016

target sounds than the (human) sound designers' were, in all objectives with the exceptions that we mention above.

8. Conclusions and future works

We focused on the application of EC to automate the task of tuning the parameters of the OP-1, a complex commercial synthesizer developed by Teenage Engineering, to replicate or approximate given target sounds.

8.1 Contributions

This work provides several contributions to the field of the preset generation for a synthesizer.

- In Section 5, NSGA-II is presented, which we deploy with a 3-objective fitness function, Gray code encoding and a modified crossover operator to preserve population diversity and enable the user to receive a small set of distinct solutions rather than a unique solution as with previous systems.
- In Section 5.4, a 3-objectives fitness function including FFT, Envelope and STFT is developed, which addresses some of the difficulties associated with the exploration of a multi-modal search space such as the OP-1 parameters space.
- In Section 5.6, a clustering method has been developed to better analyse and explore the set of final solutions. This method is based on k-mean clustering and the silhouette methodology to set the clustering size.
- In Section 6.3, an evaluation is proposed using contrived and non-contrived sounds. Trials revealed the capabilities of *PresetGen* to optimize the parameters of the OP-1 synthesizer to approximate contrived and non-contrived target sounds.
- In Section 7, we provide an empirical perceptual study that also validates the human-competitive nature of *PresetGen*. Instead of ordinal similarity, we use cardinal similarity in the experiment design.

This applied work contributes to the field of sound synthesis using an evolutionary system to find OP-1 synthesizer presets to reproduce given target sounds. Our evaluation, especially the one using contrived target sounds, will make it possible to easily compare the performances of *PresetGen* to the performances of future systems developed only for the OP-1. In the evaluation with contrived target sounds in Section 6.3, we define the notion of global fitness/distance correlation and local fitness/distance correlation. The high global fitness/distance correlation shows that our algorithm converges, on average, to the region in the preset space where the target preset is located and the low local fitness/distance correlation explains why *PresetGen* is not able to converge exactly to the target preset at the end of the optimization.

The GA system described in this work also contributes to the field of synthesizer preset generation applications of EC. *PresetGen* is based on the NSGA-II, a multi-objective genetic algorithm. This multi-objective approach (one that is not common in the field of synthesizer preset generation) has been shown to produce particularly robust results when used to find OP-1 presets to match given target sounds. Using three objectives (FFT, Envelope and STFT) instead of only one as in previous works, made it possible to solve the complex, multimodal and multidimensional optimization problem raised by real world synthesizers such as OP-1. These three objectives, combined with the intrinsic mechanisms of the NSGA-II (Non-domination sorting, diversity preservation and elitism) preserve multiple solutions located in diverse regions of the search space, therefore, to avoid a premature convergence. A modified crossover operator that prevents the recombination of two individuals with the same genotype has also been introduced to preserve diversity. This multi-objective approach also enables users to receive a set of solutions (that can use different synthesis engines, LFO or FX) rather than a unique solution, as with previous systems.

PresetGen can be used with other complex commercial synthesizers without any changes except, of course, the genotype encoding of the parameters and the synthesizer simulator.

8.2 Improvements to the system

The current system conducts the search in the parameter space of OP-1. In our implementation, we showed that certain parameter changes create sounds that are not noticeably different regarding human hearing. One could argue that parameter space can be pre-processed to create a sound search space. This sound search space would have all possible OP-1 sounds that have a noticeable difference to human hearing. Then, *PresetGen* can conduct the search in this space. However, this pre-processing creates new issues. The computational complexity of the system would increase because the OP-1's parameter space has 10^{76} possible combinations to be compared with each other.

At present, the evolution needs a large amount of computational power to determine good presets to match a given target sound. We provided the number of generations to show the computational complexity of *PresetGen* throughout this paper because we ran our experiments on a variety of computers and computer clusters. We currently evolve a population of 500 individuals over thousands of generations. A single run requires approximately 5 h on the Westgrid Bugaboo cluster with our algorithm distributed on 50 cores. However, as execution time was not a priority in this work, there are numerous optimizations of the system. For example, the population size and other GA parameters such as mutation and crossover probabilities or the stopping criteria could be adjusted. Another idea is to reduce the time complexity of extracting the three fitness objective values for every individual of the population. For instance, an optimized temporal segmentation of

Downloaded by [Simon Fraser University] at 13:21 29 July 2016

the STFT could reduce the time computation, but also give a more suitable measure for this objective. Furthermore, we can decrease the computational complexity by researching a single objective fitness function that adapts itself depending on the nature of target sounds.

FX and LFO module types were also challenging to identify for *PresetGen*. Studying an additional objective is a good idea to take the nature of these modules into account. It would also be interesting to separate the optimization of the knob parameters from the type parameters. In this perspective, using a co-evolution genetic algorithm; where one population representing the types is co-evolved with another population representing the knob parameters—seems promising.

Results of empirical evaluation with non-contrived sounds pointed out that *PresetGen* does not perform better than humans on the task of matching a non-contrived sound with impulse characteristics. We also plan to improve *PresetGen* to achieve human-competitiveness with non-contrived sounds with impulse characteristics by updating the system's fitness function and experimenting on different objectives.

8.3 Improvements to the evaluation

Our evaluation identified discrepancies in performances using target sounds of different natures. A more in-depth study might explain these differences and would allow us to improve *PresetGen*. For example, target sounds with a dynamic spectrum are more difficult to match than a stationary spectrum. Adding an objective related to this characteristic of the sound could improve the overall system performance. Our target sounds were limited to a 2 s duration. It would be interesting to study the performance of *PresetGen* for longer sounds.

It was also challenging to compare the performances of *PresetGen* to other systems in the literature. Indeed, different target sounds and performance indicators are used in the different previous works. Developing a benchmark that includes target sounds of different natures and performance indicators could make it possible to easily compare similar systems, as well as build on previous works.

Even though our experimentations during the designing phase of our system led us to switch from the canonical GA to a multi-objective GA (NSGA-II), it has not been proved formally that the NSGA-II outperforms the canonical GA. In this perspective, it would be interesting to do a direct comparison experiment to compare formally the performance of our NSGA-II system to a canonical GA.

As a future work, we also plan to expand our empirical evaluation experiment to multiple sound designers. Our initial fitness value comparisons with multiple sound designers in Section 7.3 gave encouraging results showing that *PresetGen* matched the target sounds better than (human) sound designers, generally speaking.

The *sound designer 1* commented that the synthesizer's user interface provided convenience to match target sounds. For example, to generate a plucked instrument sound, the sound designer searched the parameters starting from the presets that had names involving plucked instruments. For this reason, we plan to research the relationship between the designer's performance and the synthesizer's user interface as a future work.

8.4 Applications

A practical application of *PresetGen* would be an online platform in which a user could upload target sounds. The presets search would be evolved offline using *PresetGen* and the resulting OP-1 presets would be sent back to the user by email.

An adapted version of our GA system could also be integrated into an online OP-1 patch randomizer.[8] The idea here would be to use an interactive GA instead of an NSGA-II. The user would be asked to rank by preference the individuals in the population. These rankings would be used to select the individuals for mutation and crossover. It would also be possible to use our NSGA-II system for *background evolution* (McDermott, Griffith, & O'Neill, 2007). Here, a target sound would be loaded before any user interaction takes place. Our NSGA-II algorithm would then run in the background, attempting to match the target sound. Meanwhile, the user would interact with the system using a GUI in the *foreground*. For each generation, the individuals in the cumulative Pareto front would migrate from *background evolution* to *foreground evolution*.

PresetGen can also be useful as an educational tool in sound design. The system can point out alternative ways to match a given target sound on the synthesizer.

Working on the OP-1 optimization problem gave us numerous insights on how to solve the harder synthesizer generation problem (Macret & Pasquier, 2014). Although this problem presents a lot of similarities to the OP-1 problem, it is more complicated in the sense that PD's audio synthesis architecture is non-linear and modular. A limited number of synthesis engines, LFO and FX are accessible with the OP-1. On the other hand, PD's building blocks are fundamental synthesis components, such as oscillators or filters. It is possible to generate any synthesis engines, LFO and FX using PD. It makes the search space for the synthesis architecture subsequently more complex for PD than the OP-1. The number of input parameters is fixed in the OP-1 case, but it can vary in the PD case, making the search even more complex. Given the complexity difference, using the same optimization system for PD than for the OP-1 does not look promising. Instead, we implemented the idea of using co-evolution to separate the optimization of the synthesis architecture from the synthesis parameters with our Automatic PureData patch generation system. Limiting the number of input parameters for PD makes sense from a usability perspective. It also scales well with new promising Evolutionary techniques (such as Cartesian Genetic Programming) that can evolve graphs, the natural representation for PD patches.

[8] http://op-rand1.appspot.com/welcome.jsf

Downloaded by [Simon Fraser University] at 13:21 29 July 2016

This work brings us closer to making synthesizers more accessible to novice practitioners and helping free the musician or composer from tedious calibrations so that they can focus on the aesthetics of an artwork without losing the context.

Acknowledgements

We would like to thank Teenage Engineering, the makers of OP-1 synthesizer, for their collaboration and support. Special thanks goes to Laurent Droguet from Institut de Recherche et Coordination Acoustique/Musique (IRCAM) and Dr. Corey Kereliuk from Technical University of Denmark. We also want to thank Pr. Bernhard Riecke for his help with methodology and the statistics of the empirical evaluation experiment.

Funding

This research was funded the Natural Sciences and Engineering Research Council of Canada, and Social Sciences and Humanities Research Council of Canada.

References

Barbulescu, L., Watson, J.-P., & Whitley, L. (2000). Dynamic representations and escaping local optima: Improving genetic algorithms and local search. In *Proceedings of the National Conference on Artificial Intelligence* (pp. 879–884). Palo Alto, CA: AAAI Press.

Bozkurt, B., & Yüksel, K. (2011). Parallel evolutionary optimization of digital sound synthesis parameters. In *Proceedings of the Conference on Applications of Evolutionary Computation* (Lecture Notes in Computer Science, Vol. 6625, pp. 194–203). Berlin: Springer.

Chan, S., Yuen, J., & Horner, A. (1996). Discrete summation synthesis and hybrid sampling-wavetable synthesis of acoustic instruments with genetic algorithms. In *Proceedings of the International Computer Music Conference (ICMC)* (pp. 49–51). International Journal of Advanced Computer Science and Applications (IJACSA).

Chaudhari, M., Dharaskar, R. V., & Thakare, V. (2010). Computing the most significant solution from Pareto front obtained in multi-objective evolutionary. In *International Journal of Advanced Computer Science and Applications, 1*(4), 63–68.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Evolutionary Computation, 6*(2), 182–197.

Fortin, F., De Rainville, F., Gardner, M., Parizeau, M., & Gagné, C. (2012). DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research, 13*, 2171–2175.

Garcia, R.A. (2001). Automatic generation of sound synthesis techniques (PhD thesis). MIT, Cambridge, MA, USA.

Hinton, P.R. (2004). *SPSS explained*. London: Routledge.

Holland, J.H. (1975). *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*. Ann Arbor: University of Michigan Press.

Horner, A., & Beauchamp, J. (1996). Piecewise-linear approximation of additive synthesis envelopes: A comparison of various methods. *Computer Music Journal, 20*(2), 72–95.

Horner, A., Beauchamp, J., & Haken, L. (1993). Machine tongues XVI: Genetic algorithms and their application to FM matching synthesis. *Computer Music Journal, 17*(4), 17–29.

Jin, Y., & Branke, J. (2005). Evolutionary optimization in uncertain environments-A survey. *Journal on Evolutionary Computation, 9*(3), 303–317.

Johnson, D. (2002). A theoretician's guide to the experimental analysis of algorithms. *Data Structures, Near Neighbor Searches, and Methodology: 5th and 6th Dimacs Implementation Challenges, 59*, 215–250.

Jones, T., & Forrest, S. (1995). Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In *Proceedings of the International Conference on Genetic Algorithms* (pp. 184–192). San Francisco, CA: Morgan Kaufmann.

Lai, Y., Jeng, S., Liu, D., & Liu, Y. (2006). Automated optimization of parameters for FM sound synthesis with genetic algorithms. *In International Workshop on Computer Music and Audio Technology*,

Lobo, F., Lima, C., & Michalewicz, Z. (2007). *Parameter setting in evolutionary algorithms* (Vol. 54). Berlin: Springer Verlag.

Macret, M., & Pasquier, P. (2014). Automatic design of sound synthesizers as pure data patches using coevolutionary mixed-typed Cartesian genetic programming. In *Proceedings of the 2014 Conference on Genetic and Evolutionary Computation* (pp. 309–316). New York: ACM.

Macret, M., Pasquier, P., & Smyth, T. (2012). Automatic calibration of modified FM synthesis to harmonic sounds using genetic algorithms. In *Proceedings of Sound and Music Computing Conference* (pp. 387–394). Barcelona: SMCNetwork.

Mathieu, B., Essid, S., Fillon, T., Prado, J. & Richard, G. (2010). Yaafe, an easy to use and efficient audio feature extraction software. In *Proceedings of the International Society for Music Information Retrieval* (6pp). Canada: International Society for Music Information Retrieval.

MATLAB, (2011). MATLAB version 7.12.0 (R2011a). Natick, MA: The MathWorks Inc.

McDermott, J., Griffith, N., & O'Neill, M. (2007). Evolutionary GUIs for sound synthesis. In *International Conference on Applications of Evolutionary Computing* (Lecture Notes in Computer Science, Vol. 4448, pp. 547–556). Berlin: Springer.

Mitchell, T. (2010). An exploration of evolutionary computation applied to frequency modulation audio synthesis parameter optimisation, (PhD thesis, University of the West of England, Bristol, UK). Retrieved from http://www.teamaxe.co.uk.

Mitchell, T. (2012). Automated evolutionary synthesis matching. *Journal on Soft Computing, 16*(12), 2057–2070.

Mitchell, T., & Creasey, D. (2007). Evolutionary sound matching: A test methodology and comparative study. In *International Conference on Machine Learning and Applications* (pp. 229–234). Piscataway, NJ: IEEE.

Downloaded by [Simon Fraser University] at 13:21 29 July 2016

Moffat, D., & Kelly, M. (2006). An investigation into people's bias against computational creativity in music composition. In *The Third Joint Workshop on Computational Creativity* (6pp). Trento, Italy: Universita di Trento.

Peeters, G. (2004). *A large set of audio features for sound description (similarity and classification) in the CUIDADO project* (Technical report). Paris: IRCAM.

Riionheimo, J., & Välimäki, V. (2003). Parameter estimation of a plucked string synthesis model using a genetic algorithm with perceptual fitness calculation. *EURASIP Journal on Advances in Signal Processing, 2003*(8), 791–805.

Rocha, M., & Neves, J. (1999). Preventing premature convergence to local optima in genetic algorithms via random offspring generation. *Multiple approaches to intelligent systems* (Lecture Notes in Computer Science, Vol. 1611, pp. 127–136). Berlin: Springer.

Roth, M. (2011). A comparison of parametric optimization techniques for musical instrument tone matching. *Proceedings of the Audio Engineering Society Convention* (pp. 972–980). New York: Audio Engineering Society (AES).

Rousseeuw, P. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics, 20*, 53–65.

Schatter, G., Züger, E., & Nitschke, C. (2005). A synaesthetic approach for a synthesizer interface based on genetic algorithms and fuzzy sets. In *Proceedings of the International Computer Music Conference* (pp. 664–667)

Sivanandam, S.N., & Deepa, S.N. (2007). *Introduction to genetic algorithms*. New York: Springer.

Takala, T., Hahn, J., Gritz, L., Geigel, J., & Lee, J. (1993). Using physically based models and genetic algorithms for functional composition of sound signals, synchronized to animated motion. In *Proceedings of the International Computer MusicConference* (pp. 180–185).

Tatar, K., Macret, M., & Pasquier, P. (2015). *Experiment results*. Retrieved from http://metacreation.net/PresetGen/index.html.

Teenage engineering, (n.d.) Retrieved from http://www.teenageengineering.com/

Vuori, J., & Välimäki, V. (1993). Parameter estimation of non-linear physical models by simulated evolution-application to the flute model. In *Proceedings of the International Computer Music Conference* (pp. 402–402). The International Computer Music Association.

Wakefield, G., & Mrozek, E. (1996). Perceptual matching of low-order models to room transfer functions. In *Proceedings of the International Computer Music Conference, Barcelona, 1998* (pp. 111–113).

Wehn, K. (1998). Using ideas from natural selection to evolve synthesized sounds. In *Proceedings of the Digital Audio Effects DAFX98 Workshop* (pp. 159–167).

Westgrid - Compute Canada. (n.d.). Retrieved from http://www.westgrid.ca/ (last accessed January 2016).

Yee-King, M., & Roth, M. (2008). Synthbot: An unsupervised software synthesizer programmer. *Proceedings of International Computer Music Conference (ICMC-08)* (pp. 184–187). University of Sussex.

Yee-King, M.J. (2011). Automatic sound synthesizer programming: techniques and applications, (PhD thesis, University of Sussex, Brighton, UK). Retrieved from http://core.ac.uk/download/pdf/2710683.pdf

Downloaded by [Simon Fraser University] at 13:21 29 July 2016