QUALITY DIVERSITY FOR SYNTHESIZER SOUND MATCHING

Naotake Masuda and Daisuke Saito

University of Tokyo
Graduate School of Engineering
Tokyo, Japan
`n_masuda@gavo.t.u-tokyo.ac.jp`

ABSTRACT

It is difficult to adjust the parameters of a complex synthesizer to create the desired sound. As such, *sound matching*, the estimation of synthesis parameters that can replicate a certain sound, is a task that has often been researched, utilizing optimization methods such as genetic algorithm (GA). In this paper, we introduce a novelty-based objective for GA-based sound matching. Our contribution is two-fold. First, we show that the novelty objective is able to improve the quality of sound matching by maintaining phenotypic diversity in the population. Second, we introduce a quality diversity approach to the problem of sound matching, aiming to find a diverse set of matching sounds. We show that the novelty objective is effective in producing high-performing solutions that are diverse in terms of specified audio features. This approach allows for a new way of discovering sounds and exploring the capabilities of a synthesizer.

1. INTRODUCTION

Recently, the advent of software synthesizers has made the use of synthesizers prevalent in musical production. However, there is still great difficulty in manually adjusting the parameters of a synthesizer to obtain a desirable output. It is common for music producers to simply rely on presets, i.e. parameter settings crafted by sound designers. The full creative potential of synthesizers has been out of reach of most producers. Thus, there have been many efforts to alleviate the difficulty of using synthesizers. A common approach is *sound matching*, estimation of synthesis parameters that closely replicate the target sound.

In one of the earliest examples of sound matching, genetic algorithm (GA) was used to optimize some parameters of frequency modulation (FM) synthesis to match a target spectrum [1]. Historically, FM synthesis had been viewed as a way to create emulations of natural sounds, but finding such parameters manually required time and effort. Similarly, GA has been applied to estimate the physical modelling parameters of plucked strings [2].

More recently, there have been attempts to estimate the parameters of conventional software synthesizers. Such synthesizers feature many parameters, and include not only parameters related to the static nature of the timbre but also parameters that modulate these parameters, such as LFO rate. Techniques such as linear regression [3], multi-objective GA [4], and neural networks [5], [6], [7] have been applied to multitude of synthesizers with varying degrees of complexity. Previous work has found that it is difficult to match a target sound even if the target sound was produced by the same synthesizer [5]. This difficulty can be attributed to the complexity of the synthesis algorithm and the vastness of the search space.

One fundamental question that must be addressed is whether we actually need to match a sound with a synthesizer. One motivation for early work in FM synthesis sound matching was that FM synthesis can provide a computationally efficient replacement for realistic sounds [1]. In modern music production where samplers have become computationally inexpensive and adequately expressive, this is no longer a valid reasoning for sound matching.

We argue that the objective for sound matching is to assist the user during the sound design process. The user can query the system with a target sound that is vaguely similar to the real intention. The system should then provide an intuitive way to adjust the sound further. Some methods have been proposed to assist the user in such a way. Relevance feedback from the user can account for the gap between the intended sound and the target sound [8]. A variational autoencoder model can not only perform sound matching but also provide the user with alternative controls for a synthesizer [6].

In this paper, we solve this problem by finding solutions that are spread out in terms of audio features that the user can specify. The user can then choose the sound closest to their intention from a variety of matching sounds. Specifically, the contributions of this study are as follows:

- We show that incorporating a novelty objective can improve the performance of GA-based synthesizer sound matching. In recent evolutionary computation literature, novelty objectives have been shown to effectively maintain phenotypic diversity and improve the performance of GA in difficult domains [9]. For the first time, we apply novelty objective to the problem of synthesizer sound matching.
- We introduce the idea of *quality diversity* (QD) to the problem of synthesizer sound matching. The idea is to obtain not only the single best solution, but many interesting solutions spread across a space of audio descriptors. We show that standard GA fails to produce such diverse output, but introducing a novelty objective can help in finding a diverse set of solutions.

This paper is structured as follows. Section 2 explains the basics of GA and introduces the concept of novelty search and quality diversity. In Section 3, we explain the synthesizer sound matching method and the definition of audio novelty. In Section 4, the experiment setup details regarding the optimization algorithms and the synthesizer are described. We report and discuss the results of the experiment in Section 5 and conclude the paper in Section 6.

2. BACKGROUND

2.1. Genetic Algorithms

Genetic algorithm (GA) is a metaheuristic optimization algorithm that takes inspirations from mechanisms of biological evolution. In GA, an individual consists of a *genotype* and a *phenotype*. The genotype of an individual corresponds to the genomes of an organism and is subject to the genetic operators such as crossover and mutation. The genotype is usually an array of binary or real-valued numbers. The phenotype corresponds to the observable traits and physical form of the organism. The phenotype is what decides the fitness values of each individual.

In GA, an initial population of individuals is randomly generated. Then, the phenotype of each individual is evaluated against a problem, and each individual is assigned a fitness value. Individuals are then stochastically selected from the population, favoring those with higher fitness value. Genetic operators such as mutation and crossover are performed on the genotypes of selected individuals to create the new generation of solutions. This process is repeated for a fixed number of generations.

When applying GAs to a problem, it is important to consider the design of the genetic encoding. In binary encoding, the most traditional encoding type in GA, each genotype is an array of bits (0 or 1). While this allows for simple genetic operators and is suitable for some type of problems, it is difficult to apply this to problems with real-valued inputs. Real-valued encoding is more common in such cases, but more sophisticated crossover methods such as blend crossover or simulated binary crossover [10] should be used with real-valued encoding.

2.2. Multi-objective Genetic Algorithms

While standard GAs maximize a single fitness value, multi-objective GAs (MOGAs) optimize multiple objectives at once. This is different from simply adding multiple objective functions together. Multi-objective GAs aim to find a set of *Pareto-optimal* solutions. A solution is Pareto-optimal when it is not dominated (i.e. beaten in every objective) by any other solution. The set of Pareto-optimal solutions is called the *Pareto-front*.

NSGA-II [11] is one of the most popular algorithms for solving multi-objective problems. NSGA-II performs non-dominated sorting, where least dominated solutions are given a high rank. Binary tournament selection is used for the selection of parents in NSGA-II. More specifically, two individuals are compared, and the one with the higher rank is selected. When the two individuals are of the same rank, the crowding distances are compared, favoring the individual in less crowded area of the Pareto-front. This crowding distance acts as a diversity-preserving mechanism along the Pareto-front.

2.3. Novelty Search

In GAs, progress is measured with the fitness function, but in deceptive domains, this can misdirect the search. *Novelty search* abandons the fitness objective and searches only for novel behaviors [12]. Novelty search is surprisingly effective in solving deceptive problems like maze navigation, where fitness-based GA often leads evolution to a dead end.

Novelty of an individual is defined as how different an individual is from other individuals. This is calculated as the sparseness ρ of a point x, i.e. the average distance to its k-nearest neighbors:

$$\rho(x) = \frac{1}{k}\sum_{i=0}^{k} \mathrm{dist}(\mathbf{x}, \boldsymbol{\mu}_i), \tag{1}$$

where μ_i is the ith-nearest neighbor of $\mathbf{x}$, using the distance metric dist. Usually, Euclidean distance is used as the distance metric.

Since the phenotype itself is complex and high-dimensional, the novelty is calculated from a compact representation of the phenotype, commonly referred to as the *behavior characterization* (BC). The BC should describe salient aspects of an individual's phenotype as a vector. For example, in the case of maze navigation, it is common to use the final position of the navigating robot as the BC. By rewarding the robot for reaching new areas, the robot should eventually reach the goal.

In the most simple approach for novelty search, novelty of an individual is calculated using k-nearest neighbors in the current population. However, this is known to cause a *cycling* behavior, where the population moves from one area of the phenotype space and to a new area and back again [13]. Since the algorithm has no memory, regions already explored may be visited again in later generations.

To suppress this cycling behavior, an *archive* of individuals is kept over generations. The nearest neighbor search for novelty calculation is performed against the union of this archive and the current population. Every generation, some individuals from the current population are added to the archive. While it is common to add only the individuals with novelty score above a certain threshold value, recent study has shown that adding a fixed number of individuals randomly is just as effective in improving solution quality [14].

The effectiveness of novelty search can be attributed to the fact that novelty search asymptotically behaves as a *uniform random search process in the phenotype space* [15]. This is different from a random search in the genotype space. The phenotype space is often more compact than the genotype space and thus uniform search in the phenotype space is more effective.

In domains where the phenotype space is large, novelty search alone cannot reach an adequate solution. In such cases, a novelty objective can be combined with a fitness objective. A multi-objective GA such as NSGA-II can then be used to optimize both fitness and novelty. In difficult domains such as bipedal locomotion, using a novelty objective alongside fitness has been shown to improve solution quality compared to using fitness alone [9]. This can be attributed to the fact that the novelty objective maintains a *phenotypic diversity*, keeping many stepping stones that lead to the objective.

Phenotypic diversity should not be confused with *genotypic diversity*. In fact, much of research on diversity mechanisms in evolutionary algorithms focused on maintaining diversity in the genotype space, such as fitness sharing [16]. However, in many problems, points in the genotype space often collapse into a single phenotype. For example, in case of synthesizer sound matching, many parameter settings actually result in the same sound. In such cases, it may be more effective to explicitly maintain phenotypic diversity using a novelty objective.

So far, novelty search has seen little application to music and audio. A blog post describes a project in which novelty search was applied to Ableton Live set parameters in order to find musical inspirations [17]. However, the novelty of each individual was calculated from the genotype, not the phenotype. Thus, the creative capabilities of this project is similar to that of parameter randomization features commonly found in creative software. In fact, it would be very time consuming to render each individual and evaluate their phenotypic novelty using Ableton Live.

2.4. Quality Diversity

Quality diversity (QD) algorithms aim to find a collection of solutions that are diverse and high-performing [18]. The idea of QD followed naturally from methods optimizing both novelty and fitness, since optimizing both objectives leads to the discovery of a wide range of solutions. However, optimizing fitness still imposes a global competition among the individuals, which leads to the algorithm preferentially exploring certain regions of phenotype characteristics that gives the best fitness. This is beneficial for finding the best single solution, but it may be more beneficial to explore other regions to discover a diverse set of solutions.

Subsequent research in QD algorithms focused on exploring the entire phenotype space uniformly. Novelty search with local competition (NS-LC) replaces the fitness value with a *local competition score* (LC score) [19]. The LC score is calculated as the number of individuals with inferior fitness value in the k-nearest neighbors. If this number is high, then the individual is high-performing compared to other individuals in the same niche. This mechanism leads to the discovery of high-performing solutions in many different niches.

Another popular QD algorithm is MAP-Elites [13]. MAP-Elites uses an archive which explicitly defines the niches in the BC. The BC is discretized to form a grid, and each bin on the grid can hold a fixed number of individuals. The selection operator selects bins from the grid with uniform probability and then selects individuals from the bin to produce offspring. The BCs of the offspring are calculated, and the offspring are added to the corresponding bin.

QD algorithms have been applied to fields such as procedural content generation, where game stage designs are automatically generated [20]. In this case, the BC is the aesthetics of the stage or the moves required to beat it. The quality of a stage is how well an A* agent can beat it (i.e. playability). By generating a diverse set of solutions spread along a specified dimension, the user can easily pick an appropriate solution. To our knowledge, QD methods have yet to be applied to the field of sound design.

3. PROPOSED METHOD

3.1. Synthesizer Sound Matching

In GA-based synthesizer sound matching, the genotype is the synthesizer parameters and the phenotype is the synthesized audio, as shown in Figure 1. The genotype is an array of real-valued numbers. This is fed into the synthesizer as the synthesis parameters to produce an audio output. The fitness of the individual is calculated by how well the synthesized audio matches the target sound.

Several loss functions for measuring the match quality have been used in previous work, such as spectrogram loss [4] and mel-frequency cepstrum coefficients (MFCC) loss [5]. Preliminary experiments found that MFCC loss gave subpar matches, often resulting in sounds with pitch different from the original. This is due to the fact that MFCC only captures the overall shape of the spectral envelope. This property is ideal for applications such as speech recognition, but unsuitable for sound matching. Thus, we calculate match loss using the spectrogram.

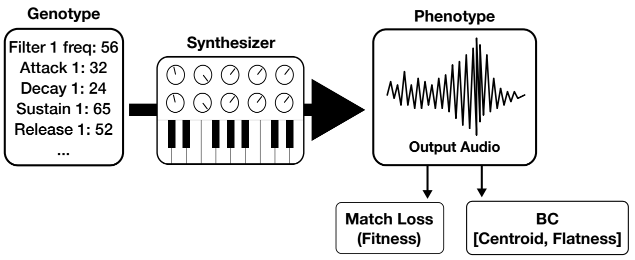

Figure 1: *The genotype-phenotype relation in synthesizer sound matching.* Expression of the genotype corresponds to playing the synthesizer. The phenotype, or output audio is used to calculate fitness and behavior characterization.

Algorithm 1 Proposed method (fitness & novelty)

```
Input: Target, Synthesizer
  Initialize Population as list of random Individuals
  Initialize Archive as empty list
  for $gen = 1, \cdots, N_{gen}$ do
      for ind ∈ Population do                    ▷ Evaluation of Individuals
          audio ← Synthesizer(ind.gtype)
          ind.fitness ← $-LSD$(Target, audio)
          ind.bc ← $BC$(audio)
      for ind ∈ Population do                    ▷ Novelty calculation
          distances ← $kNN$(ind.bc, Population ∪ Archive)
          ind.novelty ← $\frac{1}{k}\sum$ distances
      Offspring ← $SelectionNSGAII$(Population)
      for $i = 0, \cdots, N_{pop}/2 - 1$ do      ▷ Genetic operators
          ind$_a$, ind$_b$ ← Offspring[$2i$], Offspring[$2i+1$]
          Mutate ind$_a$, ind$_b$ at mutation rate $p_{mut}$
          Crossover ind$_a$ and ind$_b$ at crossover rate $p_{cx}$
      Population ← Offspring
      Add $\lambda$ Individuals from Offspring to Archive at random
```

A pseudocode of the proposed method can be written as Algorithm 1. The initial population is generated randomly by sampling uniformly from the parameter ranges. The genotypes of the individuals in the population are fed into the synthesizer as synthesis parameters to synthesize audio. STFT is then calculated for the synthesized sounds to extract the spectrogram, and the log-spectral distance (LSD) is measured between the synthesized and target audio. LSD of two power spectra is calculated as:

$$LSD(P, \hat{P}) = \sqrt{\sum_{\omega} 10 \log_{10} \left(\frac{P(\omega)}{\hat{P}(\omega)} \right)^2}. \qquad (2)$$

The average value of LSD over frames is used as a measure of quality of the match. Note that since tournament selection is used in NSGA-II, the scaling of this fitness value has no effect on the selection probability.

Then, behavior characterization (BC) of the synthesized audio is calculated from the spectrogram. Novelty of the individual is calculated by the mean of Euclidean distances in BC space from the k-nearest neighbors in the current population or the archive. NSGA-II selection is used to select the individuals, whose genotypes are then mutated and crossed-over to create the new population. From the new population, λ randomly selected individuals are added to the archive. This process is repeated for N_{gen} generations. We also implement novelty search with local competition (NS-LC) by replacing the fitness value with a local competition score.

3.2. Behavior Characterization

The novelty of an individual is calculated from not the synthesized audio itself but the BC of the audio. This representation should be compact and interpretable. We chose the following spectral features for the BC:

- **Spectral Centroid** is the center of mass of a spectrum. It has been shown to be correlated with the perceived brightness of a sound. When the spectral centroid is high, the sound is perceived as bright.
- **Spectral Flatness** describes how noise-like a sound is. A spectrum with high flatness is similar to white noise, while a spectrum with low flatness is similar to a pure tone.

The spectral features are calculated for every frame, and weighted mean with regards to RMS of the frame is used as the BC. Novelty objective encourages diversity along these features, so we expect to obtain sounds that are differing in brightness and noisiness.

4. EXPERIMENTS

4.1. Optimization Methods

We implemented standard genetic algorithm (fitness only, GA), novelty search with global competition (fitness & novelty, NS-GC), and novelty search with local competition (local competition score & novelty, NS-LC). All synthesizer parameters used in this experiment were continuous values. Thus, we used genetic operators for real-valued GA. In all algorithms, simulated binary crossover and polynomial mutation was used, with crossover probability $p_{cx} = 0.75$, mutation rate $p_{mut} = 0.05$ for each attribute, and crowding degree of 5. These values were chosen after preliminary experiments on the standard GA baseline. For standard GA, we used binary tournament selection to align with the selection mechanism of NSGA-II (as explained in Section 2.2) used in the latter algorithms. All algorithms were run for 200 generations with population size of 200 individuals.

In NS-GC and NS-LC, $\lambda = 5$ individuals from the population were added randomly to the archive every generation. The novelty is measured using $k = 15$ neighbors. We found that using local competition resulted in slow progress, due to the algorithm dropping the individuals with the best fitness from the population. Thus, we use elite selection to always keep 10 individuals from the last generation with the best fitness.

For comparison, we implement a synthesizer parameter estimator using an LSTM network similar to [5]. This network takes log-spectrograms as input and outputs the synthesizer parameters. The network consists of 2 LSTM layers with a hidden size of 256 and a dropout probability of 0.1, followed by a linear layer. This model was trained on 60,000 randomly generated parameter-audio pairs with MSE loss for the estimated parameters. Monitoring the parameter loss on the validation set, we found that this network quickly overfitted. This is due to the fact that the network tries to predict a single set of parameters from a sound, even though the same sound can be derived from different parameter setting. Already, we see that optimizing for parameter loss is problematic.

4.2. Synthesizer

To measure the effectiveness of the black-box sound matching methods, Dexed[1] and Diva[2] VSTi synthesizers were used for this experiment. Dexed is an emulation of the Yamaha DX7, an FM synthesizer with 6 operators, and Diva is a virtual analogue synthesizer. These two synthesizers were chosen because most of their parameters were controllable through MIDI CC. We expect that the sound matching will be easier for Diva, since it is a subtractive synthesizer that can be controlled more intuitively.

Certain sets of parameters were optimized, and other parameters were set to a fixed value during the experiments. For Dexed, the most notable fixed parameter is the ALGORITHM parameter, which modifies the routing of the FM operators. Automatic design of synthesizer structure is a separate problem that should be dealt with using methods like genetic programming [21]. For Diva, oscillator and filter model types were fixed, and the effects were turned off. 32 parameters used during sound matching with Diva include envelope and LFO parameters that modulate amplitude and filter cutoff, oscillator tuning and mix parameters, and filter feedback/resonance/cutoff parameters.

We tested Dexed with various numbers of parameters to optimize. The *3-ops* setting used 30 parameters, including parameters of 3 operators. The other 3 operators were turned off. Next, the *6-ops* setting used 61 parameters including parameters of all 6 operators. It can be hypothesized that optimization is more difficult when there are more parameters to optimize. Features such as pitch envelope and LFO were turned off for this experiment, as it was mostly unnecessary for matching natural sounds.

For rendering audio from VSTi, RenderMan [5] was used. RenderMan is a python package that enables control of MIDI CC parameters and recording of MIDI notes significantly faster than realtime. For each sound, a MIDI note with pitch of C4 and velocity value of 80 lasting 3 seconds was recorded for 4 seconds to capture the release of the sound. The sample rate was set to 22,500 Hz for the Dexed synthesizer and 44,100 Hz for the Diva synthesizer, since Diva did not function properly in lower sampling rates. We use an FFT size of 2048, and an STFT hop size of 1024 samples. Librosa [22] was used for calculation of spectral centroid and flatness. These values are then clipped to be within $[300, 4500]$Hz and $[-85, -30]$dB respectively. These limits were chosen according to the distribution of the typical musical sounds included in the test set. Lastly, the values were both normalized to be within $[0, 1]$ before calculating novelty.

4.3. Test Set

To evaluate our methods, we prepared two test sets: *in-domain* and *out-of-domain*. Each test set consists of 30 sounds. In-domain consists of sounds created using the same synthesizer. For creating this dataset, presets for each synthesizer were collected from the Internet and the built-in preset library. Certain parameter values were fixed to a default value to match the parameter optimization setting, Thus, it is theoretically possible to match the sounds perfectly, although previous work has found this task to be challenging for the Dexed synthesizer [5]. These presets were rendered

[1]`https://asb2m10.github.io/dexed/`
[2]`https://u-he.com/products/diva/`

Table 1: *Comparison of sound matching results.* Values shown are the mean log-spectral distance (the smaller the better) across each type of test set with different synthesizer settings. Standard deviation is noted in parentheses.

	In-domain			Out-of-domain		
	Diva	Dexed 3-ops	Dexed 6-ops	Diva	Dexed 3-ops	Dexed 6-ops
GA	11.47 (7.89)	22.22 (15.19)	30.50 (10.47)	11.73 (4.47)	36.20 (16.59)	32.93 (14.14)
NS-GC	**9.55** (5.64)	**18.71 (13.47)**	**27.91** (10.72)	**11.45** (4.35)	**32.26** (14.66)	**31.19** (14.91)
NS-LC	10.71 (6.83)	19.03 (12.98)	29.92 (11.00)	11.78 (4.48)	32.87 (14.89)	32.68 (14.71)
LSTM	22.77 (13.83)	43.45 (22.86)	70.47 (33.74)	19.52 (6.55)	55.22 (24.03)	54.73 (23.24)

with the same MIDI note used in the optimization. Out-of-domain consists of musical sounds that would actually be used as a query to the system in real usage. This includes sounds produced from a diverse range of acoustic and electronic instruments. These sounds were taken from the Nsynth Dataset [23]. The test set was extracted to be well spread out in terms of instrument type. These sounds are also in the same pitch as the MIDI note used in the optimization.

5. RESULTS AND DISCUSSION

In Section 5.1, we show the effects of novelty objective in improving the quality of synthesizer sound matching. In Section 5.2, we evaluate the sound matching evolution from a quality diversity perspective. Finally, we discuss a possible creative application of the proposed method and its challenges in Section 5.3. Further experiment results including output audio and animations of the evolution have been uploaded to the accompanying web page [3].

5.1. Sound Matching

In Table 1, we show the results of sound matching, in regards to the best solutions found by the evolution for GA-based methods. For the estimation of *in-domain* sounds, NS-GC (novelty & fitness) found significantly better solutions than standard GA in all settings, *Diva*, *Dexed 3-ops*, and *Dexed 6-ops* ($p = 0.004, p < 0.001, p = 0.006$, Wilcoxon signed-rank test). This shows that the preservation of phenotypic diversity was beneficial for sound matching. Despite only optimizing for local competition score, NS-LC seemed to perform just as well as standard GA in terms of the best solution found. For the *out-of-domain* dataset, NS-GC found significantly better solutions than standard GA in the more difficult settings, *Dexed 3-ops* and *Dexed 6-ops* ($p < 0.001, p < 0.001$, Wilcoxon signed-rank test). Standard deviation of the result is high for all settings, owing to the fact that the test set contains sounds of various characteristics. Sounds with simple timbre and fast decay tend to be easier to match and therefore result in better match scores, while longer, complex sounds result in worse match scores.

We found that LSTM performed poorly in comparison to GA-based methods. One possible cause for this is that it is hard for the model to learn properly while minimizing a loss of synthesis parameters, when the same sound can come from different synthesis parameter settings. What the model learned as the "ground-truth" synthesis parameter setting to a certain sound can be contradicted by another setting resulting in the same sound. Another problem is the quality of the training data. The training data for the LSTM model is generated by uniformly sampling in the parameter space, but randomly setting the parameters of a complex synthesizer often results in an unorthodox and non-musical sound. Using hand-crafted presets as training data can possibly improve the performance of the model, but a large collection of presets is usually unavailable for most synthesizers. However, one merit of using neural network-based methods is that once the network is trained, inference can be performed quickly. GA-based methods require playing the synthesizer to obtain evaluate the individuals, taking a much longer time (see Section 5.3 for further discussion of run time).

For the *in-domain* dataset using Dexed, the more difficult *6-ops* setting resulted in poorer match quality. This can be attributed to the increase in complexity of both synthesis algorithms and target sounds [4]. Interestingly, the more difficult parameter setting often resulted in better matches for *out-of-domain* dataset, suggesting that the optimization algorithms were able to utilize the more complex and powerful synthesis algorithms.

5.2. Quality Diversity

In a quality diversity (QD) problem setting, the objective is to obtain not only the best match, but a set of solutions spread across the specified audio features. For evaluation of QD, it is useful to discretize the behavior characterization (BC) space. We split each dimension of BC into 20 divisions to create a 2D grid with 400 bins. In a manner similar to MAP-Elites, every individual found during the evolution is assigned to a bin according to its BC. We denote the maximum match quality of individuals assigned to the ith bin as Q_i. This match quality was calculated as the reciprocal of log-spectral distance (the higher the better).

The distribution of Q_i in BC space for a typical run (out-of-domain sound, *Dexed 6-ops*) is shown in Figure 2. This illustrates how each algorithm exploited the niches in BC space. The standard GA only explored some regions in the BC space, while NS-GC and NS-LC explored a wider region and found high quality solutions in many bins. Since the target sound was an electric bass, regions with lower spectral centroid tended to have a better fitness.

A comparison of the algorithms in terms of various QD metrics (*coverage*, *exploration uniformity*, *QD score*) is shown in Figure 3. These metrics were calculated from and averaged over runs with *out-of-domain* dataset and *Dexed 6-ops* setting. In the following paragraphs, we will explain each metric in detail and discuss the performance of each algorithm with regards to these metrics.

As a measure of diversity of individuals found during the evolution, the *coverage* of the evolution is calculated as the proportion of bins with 1 or more individuals assigned. The coverage of each algorithm is shown in the leftmost column of Figure 3. Standard GA has poor coverage, failing to reach many regions of the BC

[3] `https://hyakuchiki.github.io/QDSynthWebpage/`

[4] Target sounds for the *in-domain* dataset were changed according to parameter settings, so that they can theoretically be matched perfectly by optimizing the parameters.

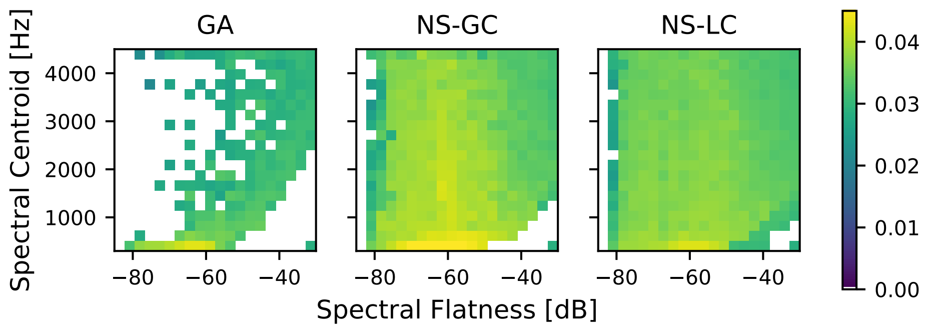

Figure 2: *Comparison of quality distribution from a typical run.* Lighter colors indicate that high-performing individuals were found in the bin. Bins with no assigned individual were colored white. The target sound was an electric bass sound from the NSynth dataset.

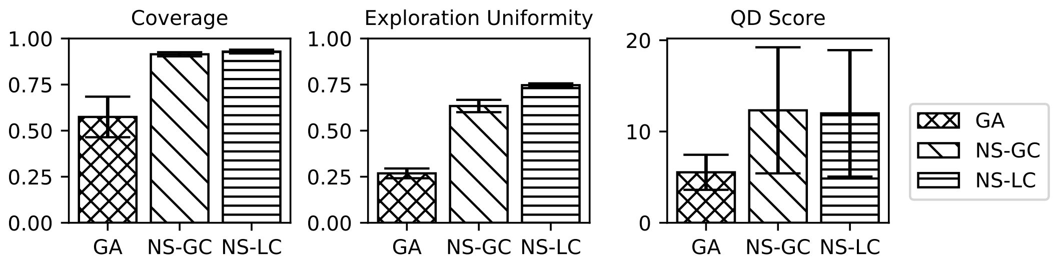

Figure 3: *Comparison of QD metrics.* Coverage values close to 1 indicate that the algorithm explored most of the bins in the BC space. For exploration uniformity, values close to 1 indicate that the individuals found during the evolution follow a uniform distribution in the BC space. Higher QD-scores indicate that the evolution was able to find a diverse set of high-performing solutions.

space, while algorithms with novelty objective reach most of the BC space.

To gain insights about the dynamics of the evolution, we analyzed the exploration uniformity, as introduced in [14]. Let φ be the set of all individuals found during a run and I_i be the number of individuals assigned to ith bin. The exploration uniformity $U(\varphi)$ is calculated from Jensen-Shannon divergence between the distribution $\boldsymbol{X}_\varphi$ of all individuals and the uniform distribution $\boldsymbol{Y}$ in the BC space:

$$U(\phi) = 1 - JSD[\boldsymbol{X}_\varphi || \boldsymbol{Y}], \tag{3}$$

where

$$\boldsymbol{X}_\varphi = \left(\frac{I_1}{|\varphi|}, \cdots, \frac{I_{400}}{|\varphi|} \right), \tag{4}$$

$$\boldsymbol{Y} = \left(\frac{1}{400}, \cdots^{\times 400} \right). \tag{5}$$

The exploration uniformity of each algorithm is shown in the center column of Figure 3. The results show that by removing global selection pressure, NS-LC explores the BC space more uniformly than NS-GC.

Finally, QD-score is often used in QD literature as a single measure of both quality and diversity. QD-score is calculated as the sum of maximum quality achieved in each bin:

$$\text{QDScore} = \sum_{i=1}^{400} Q_i, \tag{6}$$

where $Q_i = 0$ for bins with no assigned individual. The QD-scores of each algorithm is shown in the rightmost column of Figure 3. While standard GA can find a single high quality solution, it fails to find a diverse set of solutions.

Surprisingly, NS-LC did not achieve a higher QD-score than NS-GC, despite having higher exploration uniformity. This result can be explained by the *alignment* of the BC with the quality. A BC is said to be highly aligned with the quality, when certain regions in the BC space correspond to a narrow range of fitness values [18]. The BC in this experiment is fairly aligned with the match quality. If a sound has the same BC values as the target sound, the spectrograms are similar and the match quality tends to be high. When the alignment of BC is high, finding novelty will help the evolution find regions in the BC space with higher fitness. This can perhaps explain the results in Section 5.1, where the novelty helped in finding solutions with high fitness. On the other hand, only a certain region can achieve the highest fitness values, so it may be better to exploit that region to achieve a high QD-Score. We can speculate that NS-GC was the best balance between the exploitative and exploratory behavior in the case of sound matching.

In most application of QD, the BC is unaligned with the quality, since the dimensions we want to find diversity in is often unrelated to the quality. However, in the case of sound matching, the BC is somewhat aligned with the match quality, because the target sound can also be assigned to a point in the BC space. Having the same BC values as the target sound leads to better fitness score. We argue that is still useful to find diversity in this BC space because

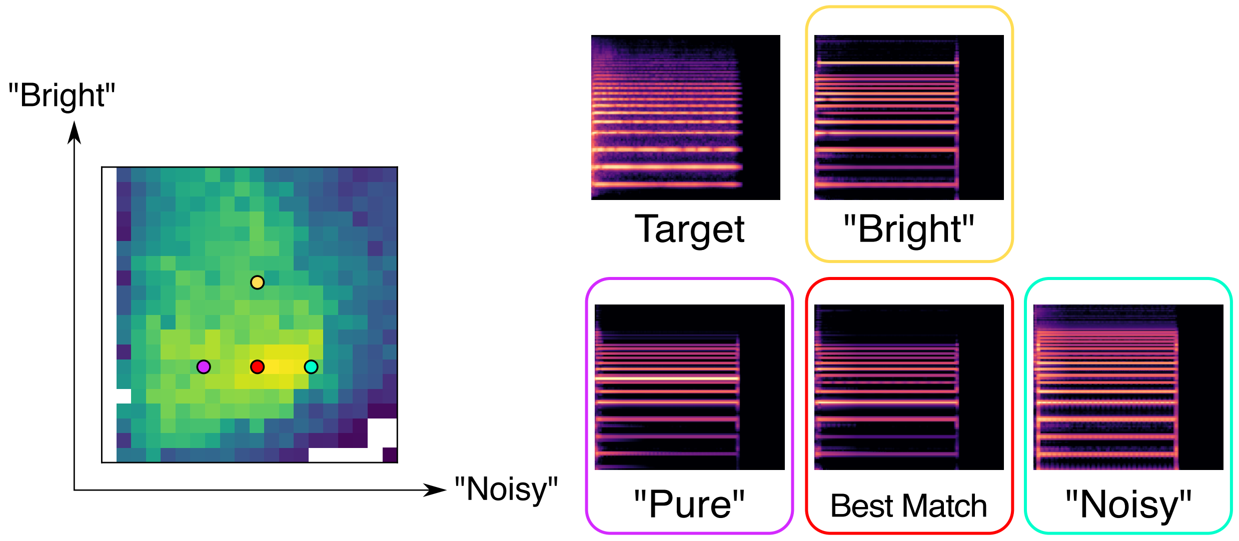

Figure 4: *Sounds contained in the BC map obtained from a typical run.* The x axis is the spectral flatness which corresponds to the "noisiness" of a sound, and the y axis is the spectral centroid which corresponds to the "brightness" of a sound. The position of the best matching individual is marked in red. We display the waveform and melspectrogram of individuals with lower spectral flatness ("Pure", purple), higher spectral flatness ("Noisy", cyan), and higher spectral centroid ("Bright", yellow) and mark the positions in BC space. In this run, NS-GC was used with the *6-ops* parameter setting to match a trumpet sound.

the target sound may differ from the user's true intentions.

5.3. Creative Applications

We propose that the product of each run is not just the best match or the final population, but all the individuals found during the population. As shown in Section 5.2, a map of BC space can be used to visualize the spread of individuals in terms of the specified audio features. We visualize the BC map and the sounds it contains in Figure 4. Bins away from the best matching individual in the BC map also holds individuals with similar structure, but differing in the spectral features specified as the BCs. For example, the "Noisy" individual is similar to the best match but features more partial, and in the "Pure" individual, a single partial is emphasized. When utilizing this method to assist sound design, a user can audition the best individuals in each bin to find sounds similar to the query sound but with a certain quality.

One challenge for applying the proposed method to a practical application is the run time. Matching a sound with 200 generations, population size of 200, and audio length of 4.0s using Dexed took about 20-25 minutes on a standard desktop CPU (Intel i7-6700). Diva took approximately an hour with the same setting, as it could not be run in lower sample rates and features computationally expensive analogue emulation. This run time may be considered as reasonable, considering that previous work in GA-based sound matching took several hours to match a single sound using a computing cluster [4]. Still, this is prohibitively long for interactive use in a creative application.

Profiling of the program revealed that approximately 95% of the elapsed time was spent on synthesizer rendering, while the calculation of novelty added little processing time. Thus, in order for GA-based synthesizer sound matching to be useful, the synthesis process needs to be optimized.

Since RenderMan did not allow for hosting of multiple VSTis, the rendering of the population was not done in parallel. A VST host software capable of hosting multiple plugins must be developed to speed up the rendering. Ideally, the synthesizer software itself should be optimized to allow for a faster batch rendering of presets. Alternatively, the parameters of the evolution can be optimized further for faster convergence, and the evolution can be ended early when a stopping criteria is met. Finally, surrogate models are often used as a way to alleviate the need for computationally expensive evaluations [24]. A surrogate model for the synthesizer can potentially approximate the fitness and the behavioral characteristics of an individual.

6. CONCLUSIONS

In this paper, a genetic algorithm with novelty objective was applied to the problem of synthesizer sound matching. Synthesized sounds were characterized by spectral features, and the novelty objective encouraged diversity along these features. We showed that the novelty objective can improve the performance of sound matching. We also introduced a quality diversity (QD) approach to the problem of synthesizer sound matching, seeking out a set of solutions that are high-performing and diverse. The novelty objective was shown to be effective in exploiting many niches in the phenotype space. This QD approach allows for a novel, intuitive way to discover sounds and explore the capabilities of a synthesizer.

7. REFERENCES

[1] Andrew Horner, James Beauchamp, and Lippold Haken, "Machine Tongues XVI: Genetic algorithms and their appli-

cation to FM matching synthesis," *Computer Music Journal*, vol. 17, no. 4, pp. 17–29, mar 1993.

[2] Janne Riionheimo and Vesa Välimäki, "Parameter estimation of a plucked string synthesis model using a genetic algorithm with perceptual fitness calculation," *EURASIP Journal on Applied Signal Processing*, vol. 8, pp. 791–805, 2003.

[3] Katsutoshi Itoyama and Hiroshi G. Okuno, "Parameter estimation of virtual musical instrument synthesizers," in *Proceedings of the 40th International Computer Music Conference*, 2014, pp. 1426–1431.

[4] Kıvanç Tatar, Matthieu Macret, and Philippe Pasquier, "Automatic synthesizer preset generation with PresetGen," *Journal of New Music Research*, vol. 45, no. 2, pp. 124–144, 2016.

[5] Matthew John Yee-King, Leon Fedden, and Mark D'Inverno, "Automatic programming of VST sound synthesizers using deep networks and other techniques," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 2, pp. 150–159, apr 2018.

[6] Philippe Esling, Naotake Masuda, Adrien Bardet, Romeo Despres, and Axel Chemla-Romeu-Santos, "Universal audio synthesizer control with normalizing flows," in *Proceedings of the 22nd International Conference on Digital Audio Effects*, 2019.

[7] Oren Barkan and David Tsiris, "Deep synthesizer parameter estimation," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 2019, pp. 3887–3891.

[8] Mark Cartwright and Bryan Pardo, "SynthAssist: Querying an audio synthesizer by vocal imitation," in *Proceedings of The International Conference on New Interfaces For Musical Expression*, 2014, pp. 363–366.

[9] Joel Lehman, Kenneth O. Stanley, and Risto Miikkulainen, "Effective diversity maintenance in deceptive domains," in *Proceedings of the 2013 Genetic and Evolutionary Computation Conference*, 2013, pp. 215–222.

[10] Kalyanmoy Deb and Ram Bhushan Agrawal, "Simulated binary crossover for continuous search space," *Complex Systems*, vol. 9, pp. 1–34, 1994.

[11] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[12] Joel Lehman and Kenneth O. Stanley, "Abandoning objectives: Evolution through the search for novelty alone," *Evolutionary Computation*, vol. 19, no. 2, pp. 189–223, 2011.

[13] Jean-Baptiste Mouret and Jeff Clune, "Illuminating search spaces by mapping elites," *arXiv preprint arXiv:1504.04909*, 2015.

[14] Jorge Gomes, Pedro Mariano, and Anders Lyhne Christensen, "Devising effective novelty search algorithms: A comprehensive empirical study," in *Proceedings of the 2015 Genetic and Evolutionary Computation Conference*, 2015, pp. 943–950.

[15] Stephane Doncieux, Alban Laflaquière, and Alexandre Coninx, "Novelty search: A theoretical perspective," in *Proceedings of the 2019 Genetic and Evolutionary Computation Conference*, 2019, pp. 99–106.

[16] Samir W Mahfoud, *Niching Methods for Genetic Algorithms*, Ph.D. thesis, University of Illinois at Urbana-Champaigne, 1995.

[17] Samim Winiger, "Musical novelty search. Evolutionary algorithms + Ableton Live," `https://medium.com/@samim/musical-novelty-search-2177c2a249cc`, 2017, Accessed: 2021-02-10.

[18] Justin K. Pugh, Lisa B. Soros, and Kenneth O. Stanley, "Quality diversity: A new frontier for evolutionary computation," *Frontiers Robotics AI*, vol. 3, no. 40, pp. 1–17, 2016.

[19] Joel Lehman and Kenneth O. Stanley, "Evolving a diversity of creatures through novelty search and local competition," in *Proceedings of the 2011 Genetic and Evolutionary Computation Conference*, 2011, pp. 211–218.

[20] Ahmed Khalifa, Scott Lee, Andy Nealen, and Julian Togelius, "Talakat: Bullet hell generation through constrained Map-Elites," in *Proceedings of the 2018 Genetic and Evolutionary Computation Conference*, 2018, pp. 1047–1054.

[21] Ricardo A. Garcia, "Automating the design of sound synthesis techniques using evolutionary methods," in *the COST G-6 Conference on Digital Audio Effects (DAFX-01)*, 2001, pp. 1–6.

[22] Brian McFee et al., "librosa/librosa: 0.8.0," `https://doi.org/10.5281/zenodo.3955228`, July 2020.

[23] Jesse Engel, Cinjon Resnick, Adam Roberts, Sander Dieleman, Douglas Eck, Karen Simonyan, and Mohammad Norouzi, "Neural audio synthesis of musical notes with WaveNet autoencoders," in *Proceedings of the 34th International Conference on Machine Learning*, 2017, pp. 1068–1077.

[24] Adam Gaier, Alexander Asteroth, and Jean Baptiste Mouret, "Data-efficient exploration, optimization, and modeling of diverse designs through surrogate-assisted illumination," in *Proceedings of the 2017 Genetic and Evolutionary Computation Conference*, 2017.