Evolutionary GUIs for Sound Synthesis

James McDermott[1], Niall J.L. Griffith[1] and Michael O'Neill[2]

[1] Dept. Computer Science and Information Systems, University of Limerick
[2] NCRA, University College Dublin
jamesmichaelmcdermott@gmail.com niall.griffith@ul.ie m.oneill@ucd.ie

Abstract. This paper describes an experiment carried out to determine which, among several possible evolutionary and non-evolutionary sound synthesizer graphical user interfaces, is the most suitable for the task of matching a target sound. Results show that standard and new varieties of evolutionary interface are competitive with a standard non-evolutionary interface, achieving better results in some situations and worse in others. Subjects' comments suggest a preference for a new type of evolutionary interface, presented here, which allows faster audition of the population, avoiding the need for time-consuming fitness evaluation of poor-quality sounds.

1 Introduction

Evolutionary Computation (EC) has been applied by several authors to the problem of setting sound synthesizer parameters, using both automatic [1], [2], [3], [4], and interactive [5], [6], [7] EC methods. Little or no research has been reported on controlled experiments comparing Interactive EC (IEC) synthesizer GUIs (Graphical User Interfaces) with non-evolutionary synthesizer GUIs. This is one aim of this paper: the other is to introduce and study a novel IEC GUI using "sweeping".

Typically, software sound synthesizers are "played" by saved performance files or by MIDI instruments; these methods determine the choice of notes, their volumes and lengths, and sometimes a few other aspects of performance. Controlling these is usually a matter of learning to compose using a sequencer or to perform using a keyboard. Synthesizers also expose a number of continuously-variable *parameters* which affect the character of the emitted sound; controlling these does not require traditional virtuosity but does require understanding of their individual purposes in the synthesizer algorithms, a knowledge of the sound character being aimed for, and a great deal of persistence. This last fact provides the motivation for this work: IEC has the potential to make control of sound synthesis parameters much easier and more intuitive, and to remove the requirement for understanding of underlying synthesis algorithms.

1.1 Existing Work

IEC is EC driven by human evaluations of fitness: it can function as a way to avoid the problem of defining explicit fitness functions for hard-to-define goals; several authors have applied IEC to sound synthesis.

A typical IGA synthesis system is described by Johnson [6]. Here, genomes are floating-point arrays, which are translated into sounds by (i.e. serve as input to) the CSound *FOF* synthesizer. The user interface consists of buttons (to hear the sounds) and sliders (to assign them fitness values). After evaluating each generation, the user clicks an "evolve" button, causing the next generation to be created using mutation and crossover.

Genophone [7] is a complex interactive system, in which a dataglove, an evolutionary software interface, and a MIDI keyboard and synthesizer are used together. Evolution takes place at the levels of synthesis parameters and of performance parameters: thus the user awards highest fitness scores to individuals which produce desired sounds as well as desired performance mappings for the dataglove. The overall process is "exploratory rather than goal orientated; it is not designed to satisfy a priori sound specifications" [7].

MutaSynth [5] is an IGA application which can be applied to different synthesis engines via MIDI control, and also controls the evolution of score material. It has been used as a publicly-accessible installation, requiring that it be usable and controllable even by very casual users.

This research is successful in proving that IEC can be applied to sound synthesis, hiding the low-level details of synthesis from a user who prefers to think in aesthetic and perceptual terms. However there has been no attempt to quantitatively compare IEC interfaces with traditional ones.

Takagi writes that IEC applications suffer from a *fitness evaluation bottleneck* [8] – the fact that usually a human will be orders of magnitude slower than a computer in evaluating an individual's fitness. This means that the large population sizes and large numbers of generations, typical of automatically-driven EC, are infeasible – and so high-quality solutions are less likely to be found. Takagi discusses several ways in which researchers attempt to improve IEC systems: one of these is combination of interactive with non-interactive evolution. We have implemented new evolutionary interfaces which combine this technique with the idea of *sweeping*, or user-controlled interpolation.

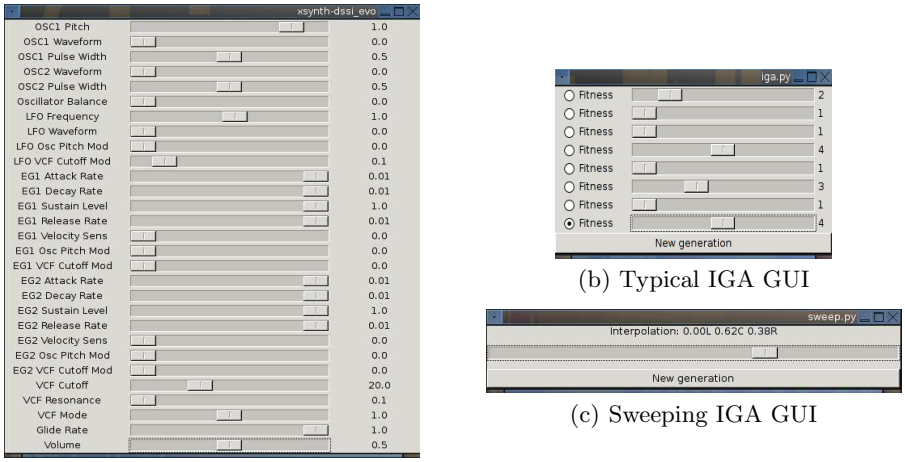
In the next section, we describe these ideas and the GUIs we have implemented and tested. In the following sections, we describe the protocol for the experiments, results and analysis, and finally conclusions and future work.

2 Software Implementation

Our experiment compares four interfaces we have implemented for the Xsynth-DSSI synthesizer [9]:

GUI 0, "Sliders", is a standard non-evolutionary GUI (Fig. 1(a)), in which each parameter is set manually by the user using a slider. Each parameter's name and current value are displayed as text next to its slider. This GUI is intended as a control.

GUI 1, "IGA", is a plain IEC GUI (Fig. 1(b)), in which a population of sounds are available for evaluation. For each sound, the user sees a radio button and a slider. The radio button selects the given sound for playing, and the



(a) Non-evolutionary GUI

(b) Typical IGA GUI

(c) Sweeping IGA GUI

Fig. 1.

slider determines its user-awarded fitness (this is Takagi's "discrete fitness value input method" [8]). A "New Generation" button allows the user to declare that fitness evaluations are finished for this generation: at this point, a new generation is made using one-point crossover with probability 1.0 and Gaussian mutation of deviation width 0.75 with probability 0.3 (these values, higher than usual, were chosen on the basis of trial experiments in which increased diversity and variation between generations was found to be necessary). This GUI approximates one version of the state of the art in IEC sound synthesis applications, as used by Johnson [6]. Of the three existing systems mentioned in Sect. 1.1, only Johnson's is directly comparable with our goals, and sufficiently well-specified to allow re-implementation. It directly controls a synthesizer and nothing else. *Genophone* involves a dataglove and the learning of performance mappings, which are not present in our work. *MutaSynth* involves the evolution of both synthesis parameters and scores. Therefore, we have chosen Johnson's interface to be re-implemented as a control.

GUI 2, "Sweep", is a sweeping-style IEC GUI (Fig. 1(c)), in which a single slider is available to the user. This GUI is a contribution of this paper. Three discrete sounds L, C and R can be accessed by placing the slider at its leftmost, centre, and rightmost points. Sounds intermediate between L and C, and between C and R, can be accessed by moving the slider to intermediate points, as follows: when the slider is between L and C, at a distance x from L, the emitted sound X has parameters $X_i = L_i + x(C_i - L_i)$ (in fact, parameters are first mapped linearly or logarithmically, as appropriate, to the interval $[0, 1]$; then X is formed; finally it is mapped back to the true parameter space). The resulting sound can be thought of as a mixture of L and C proportional to the slider's nearness to those points. A similar arrangement holds for C and R.

Any pair of points close together on the slider will also be close in the parameter space, and will usually be quite similar in the sound space. Moving the slider thus usually results in quite a gradual change of the emitted sound over the majority of the slider's range (previous work [10] has supported this claim computationally, and it is also borne out by experience with the GUI). This arrangement allows the user to quickly audition a large number of individuals with a single mouse-gesture, focussing in on the most interesting areas, and wasting no time on awarding fitness values to poor-quality sounds.

A "New Generation" button allows the user to declare that a new generation is required: the current individual is retained as point C and new individuals are randomly generated for points L and R (alternatives including the generation of L and R by mutation from C were trialled in pilot experiments, but it was again found that greater diversity was required).

The "Sweep" operator thus allows the user to hand-control an interpolation at the genetic level between pairs of individuals. This does not violate the IEC principle that users should not need to understand the function of genes, since individual parameters are not exposed. However it certainly does violate any analogy with real-world evolution. It is comparable to the (non-interactive) *morphing* operator used in MutaSynth [5]. The evolutionary mechanism can also be compared with a $(1, 3)$ Evolutionary Strategy (ES) [11].

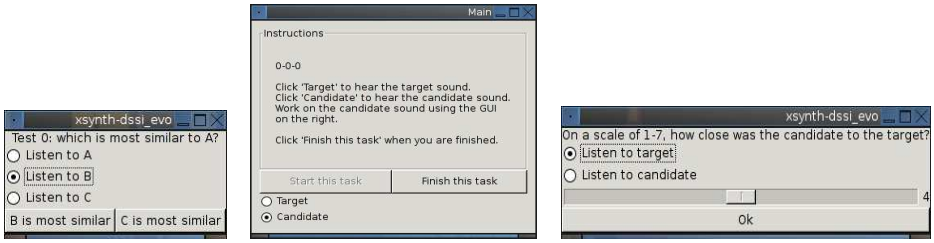
GUI 3, "Sweep with background evolution", is a sweeping-style IEC GUI (Fig. 1(c) again), augmented by background evolution. Here, a target waveform is loaded before any user interaction takes place. An automatic EC process then runs in the background, attempting to match the target waveform using a fitness function based on measurement of Attribute Distance between target and candidate sounds (see [10]). Meanwhile, the user interacts with the system as for GUI 2. In this case, the "New Generation" button indicates that the current individual is to be retained as point C; a new individual is to be randomly-generated for point L; and the best individual found so far by the background process is to be used for point R. It can happen that the user requests a new generation before the background process has found an improvement on the previous best individual: in this case a randomly-generated individual is used instead (in test experiments this is found to happen quite rarely). This is a type of Deme GA, in that at each generation one individual "migrates" from background to foreground. No migration in the opposite direction takes place.

The subjective impression of the first author is that this method does succeed in providing "raw materials" at point R which often are better than the randomly-generated sounds provided by GUI 2. It is useful in the real-world situation where a user already has a sound file exhibiting some desired characteristics, but wishes to re-synthesize to gain flexibility in pitch, loudness, duration, or timbre. It can be thought of as a way to "put knowledge in" to the system, and to exploit the complementary abilities of human and machine. However its use in this experiment is unrealistic in that here the target of background evolution is known to be exactly available using the synthesizer and is exactly the user's target.

3 Experimental Setup

3.1 Preliminary Discrimination Tests

Each user began by undertaking a series of 10 "triangle tests", in each of which the task was to listen to three sounds, A, B and C, and determine which of B and C was closest to A. In each triple of sounds, either B or C was in fact identical to A, while the other was a slightly mutated version of A. There was thus an objectively right answer to each triangle test. The purpose of this test was to gather data on how good subjects were in discriminating between sounds. The GUI for this test is shown in Fig. 2(a).



(a) Triangle Test GUI (b) Main Control GUI (c) User Rating GUI

Fig. 2.

3.2 Subjects

There were 20 subjects altogether, ranging in age between 23 and 45. There were 8 females and 12 males; in terms of synthesizer expertise, there were 6 beginners, 7 intermediate and 7 advanced, as classified by subjects themselves. Several were participants in or graduates of a one-year postgraduate level taught course in music technology.

Subjects were allowed to spend as long as they wanted, but in order to prevent them becoming fatigued or rushing the final experiments, they were advised to spend about 3-6 minutes on each task. The 8 tasks were presented in pairs, by GUI. Four different orderings of the pairs were used to avoid bias through learning or fatigue.

3.3 Target Sounds

Two target sounds were chosen from among the synthesizer's built-in preset sounds. This ensured that the target sounds were achievable using the given synthesizer, and were "desirable" sounds. They were: a percussive Xylophone-like sound (Target 0) and a typical "synth strings" sound (not a good imitation of real strings), with slow attack and release sections, and very slightly out-of-tune oscillators (Target 1). These sounds represent very different areas of the sound space.

3.4 Main Task

Figure 2(b) shows the GUI for the main part of the experiment. This GUI allowed subjects to start or finish the current task, and during the task, to switch the synthesizer back and forth between playing the target sound and the current candidate sound. In either case the sound was triggered every 1.5 seconds by a MIDI sequencer, and turned off 1 second later. On clicking the "Start this task" button, the GUI (0-3) for the current task was shown, allowing users to modify the candidate sound. On choosing to "Finish this task", subjects were presented with the User Rating GUI, shown in Fig. 2(c): here, they again listened to the target and candidate sounds, and awarded a score indicating how good a match had been achieved. After all tasks were finished, subjects filled in a short questionnaire.

4 Results and Analysis

A preliminary analysis of log files revealed that 4 subjects had, despite their instructions and trial period, failed to evolve through more than 0 generations in most of the evolutionary tasks (GUIs 1-3). This must firstly be taken as an indication that these simple GUIs are not as obvious in their function as intended. For statistical analysis, the data generated by these 4 subjects was eliminated from the dataset.

Subjects' scores for the triangle tests were all very high: it seems to have turned out to be much easier than intended (and easier than expected, on the basis of a short pilot experiment). Every subject scored either 9 or 10 out of 10. This indicates that at least at the beginning of the experiment, subjects were taking the experiment seriously and attempting to give the correct answers, and it indicates that all subjects were capable of discriminating between different sounds. The purpose of the triangle tests was to differentiate among subjects according to their ability in this regard, but since all subjects were so successful, this analysis can not be carried out.

For each task, the Subject ID, GUI, Target, Task Ordering, Time Taken and User Rating were recorded. The distance between the target sound and the sound produced by the subject was measured in three ways:

Attribute Distance works by extracting a set of 40 timbral, perceptual, and statistical attributes [12] from the sounds (via digital signal processing methods). Each attribute is then mapped, linearly or logarithmically as appropriate for perceptual reasons, from its true range to the range $[0, 1]$. The distance between two values for an attribute is then the absolute value of their difference, and an overall distance between a pair of sounds is defined by averaging these individual distances.

DFT Comparison works by dividing each sound into overlapping windows, for each window taking the Discrete Fourier Transform, and finally comparing the power in corresponding transform bins. This method is the most commonly-used in the EC synthesis literature.

Parameter Distance compares not sounds but the parameter settings which give rise to them. Each parameter is mapped from its true range to the interval $[0, 1]$, again either linearly or logarithmically as appropriate. Two sounds are compared by taking the average of the absolute value of the differences of individual mapped parameters.

4.1 Main Results

Table 1 shows the results of a two-way repeated measures ANOVA for each of User Rating, Time Taken, and Attribute Distance, where the two factors are GUI and Target.

Table 1. Average results for User Rating (user's satisfaction with the match, on a scale of 1-7; higher is better), Time Taken (in seconds; lower is better) and Attribute Distance (from target sound to achieved sound, on a unit scale; lower is better), analysed by GUI and by target sound.

		GUI 0	GUI 1	GUI 2	GUI 3	All GUIs
User Rating (1-7)	Target 0	3.44	3.06	2.50	3.06	3.02
	Target 1	3.87	4.12	4.50	4.56	4.27
	Both Targets	3.66	3.59	3.50	3.81	3.64
Time Taken (s)	Target 0	475	455	271	227	357
	Target 1	273	292	116	135	204
	Both Targets	374	374	194	181	281
Attribute Distance	Target 0	0.14	0.18	0.18	0.17	0.17
	Target 1	0.08	0.13	0.09	0.07	0.09
	Both Targets	0.11	0.16	0.13	0.12	0.13

There are no statistically significant differences in User Rating between GUIs ($F = 0.38$; $p > 0.1$). However, there are significant differences by Target ($F = 34.8$; $p < 0.001$) and by GUI against Target ($F = 3.14$; $p < 0.05$). In particular the Slider GUI (0) received the best ratings for Target 0, but the two Sweep GUIs (2 and 3) received the best ratings for Target 1. This result is reflected in the statements by two expert synthesizer users that the Sweep interfaces were more appropriate for timbral matching (the main difficulty in Target 1), while the standard GUI (0) was more appropriate for envelope matching (the main difficulty in Target 0). Thus there is evidence that the Sweep interfaces are useful in particular circumstances.

Users spent significantly less time working with the Sweep GUIs (2 and 3) than the other two ($F = 12.1; p < 0.001$). This is a positive result, since achieving the same quality of match in a shorter time benefits both the serious and the casual user.

There are significant differences in Attribute Distance by GUI ($F = 10.8; p < 0.001$): the Sliders interface (0) gave the lowest Attribute Distances (performs

best), while the IGA (1) was worst: the two Sweep interfaces were in between. Again, these results are differentiated strongly by target: for target 0, Sliders was best and all others equally bad; for target 1, IGA was worst and all others about equally good. There were no significant differences in User Rating by Task Ordering ($F = 1.24$; $p > 0.1$).

One of our motivating hypotheses was that the Sweep interfaces would be more suitable for novices. Although this hypothesis is partly confirmed by users' responses to the questionnaire, results show that there were no statistically significant differences in the User Ratings by GUI and Synthesizer Experience Group ($F = 1.40$; $p > 0.1$).

The results show significant differences in User Rating ($F = 34.8$; $p < 0.001$), Attribute Distance ($F = 217.0$; $p < 0.001$), and Time ($F = 12.1$; $p < 0.001$), by Target. This indicates that one target is more difficult than the other. This can be confirmed using the method of random sampling: we have generated 100 sounds at random in the parameter space and for each, calculated its Attribute Distance to each of the two targets. The distance to target 0 was found to be greater ($p < 0.001$): the mean Attribute Distances were 0.185 and 0.140 respectively. A random point in the Parameter space is likely to be closer to target 1. This implies that the map from Parameter to Attribute space is denser in the area of target 1 than in that of target 0, so in a sense target 0 is more difficult to find. It is likely that variations in the density of this map are characteristic of the synthesizer; this is a key issue in the problem of target-matching. Future work could use this insight in to guide the generation of new individuals in interactive algorithms.

4.2 Qualitative Analysis

In the post-test questionnaire, subjects were invited to give any comments they wished on the GUIs and experiments. Of the 9 subjects who chose to express a preference, six said that one of the Sweep GUIs was the best. One subject remarked "There was a sense of progression with the sweep GUI, whereas the sliders GUI didn't have that."

Two advanced synth users remarked that individual parameters (i.e. a standard interface) were best for setting time envelopes, while the Sweep interface was best for setting timbral aspects of the sound. This seems to be partly due to expert users finding that the evolutionary algorithms (GUIs 1-3) did not provide the necessary variation in time envelope "raw material". One further remarked that it was possible to form a mental picture of the sound's time envelope, and then to match this against a representation of the desired sound's time envelope, but that timbre resisted this mental representation.

Several users asked for features to be added, and in particular a "back" button or a save facility, in the case of the evolutionary interfaces. (These were considered during the design phase, but rejected as imposing too much interface complexity on subjects.) Often the same subjects expressed frustration that they had achieved quite a good match only to lose it in the next generation. This applies particularly, but not exclusively, to the IGA GUI (1), since the Sweep

GUI retains the "best" sound from the previous generation. One user also asked for a "reset to default" button, in the case of the standard synth interface.

In response to the potential objection that novices could quickly be taught to become intermediate synthesizer users, and thus proficient in the use of a typical synthesizer interface, we note that of the several subjects who were graduates of at least a one-year graduate-level course in music technology, some still regarded themselves as no more than intermediate-level synthesizer users and in some cases remarked that the synthesizer parameters were confusing or that they didn't understand what they did.

4.3 Correlations among Measures of Success

Table 2 gives the Pearson's product-moment correlation (and associated 95% confidence interval) between User Rating and several other measures of performance. Parameter Distance and Attribute Distance are quite strongly negatively correlated with User Rating. This lends some support to the use of Attribute Distance as a fitness function for automatic evolution, and to the use of Parameter Distance as a measure of evolutionary success for experimental use.

Table 2. Correlations between User Rating and other measures of success

Measure	Pearson's correlation	Confidence interval	Significance
Parameter Distance	-0.32	[-0.47, -0.16]	*
Attribute Distance	-0.51	[-0.63, -0.37]	*
DFT Distance	0.17	[-0.00, 0.33]	
Time	-0.18	[-0.34, -0.01]	

5 Conclusions and Future Work

We have introduced and studied a novel IEC GUI using the technique of "sweeping". Overall, the Sweep interfaces with and without background evolution have been shown to be competitive with and in some ways better than the other interfaces, as judged by User Ratings, Attribute Distances, and Time spent per task.

Although the technique of background evolution has not been shown to provide a statistically significant improvement in performance, each of User Ratings, Times, and Attribute Distances are in almost every category slightly better for GUI 3 than GUI 2. As noted above, GUI 3 represents an idealised situation: for these experiments, the target is exactly available using the given synthesizer, and a perfect recording of the target is available. Real-world use would further diminish any improvement due to background evolution.

The correlations between Attribute Distance and User Rating, and between Parameter Distance and User Rating, have application to the design and testing of automatic search algorithms.

Several modifications are suggested by the data and by user comments: extra features such as a "back" button or a save facility; modifications to the method of choosing the endpoints for the Sweep interface; and the extension of the Sweep interface to the 2-dimensional case.

6 Acknowledgements

The first co-author gratefully acknowledges the guidance of his co-authors and supervisors; many thanks also to Brian Sullivan and Dr. Jean Saunders, Statistics Consulting Unit, University of Limerick, and Dr. Fred Cummins, University College Dublin. The first co-author is supported by IRCSET grant no. RS/2003/68.

References

1. Horner, A., Beauchamp, J., Haken, L.: Machine tongues XVI: Genetic algorithms and their application to FM matching synthesis. Computer Music Journal **17**(4) (1993) 17–29
2. Garcia, R.A.: Growing sound synthesizers using evolutionary methods. In Bilotta, E., Miranda, E.R., Pantano, P., Todd, P., eds.: Proc. ALMMA 2001: Artificial Life Models for Musical Applications Workshop (ECAL). (2001)
3. Mitchell, T.J., Sullivan, J.C.W.: Frequency modulation tone matching using a fuzzy clustering evolution strategy. In: Audio Engineering Society 118th Convention. (2005)
4. Riionheimo, J., Välimäki, V.: Parameter estimation of a plucked string synthesis model using a genetic algorithm with perceptual fitness calculation. EURASIP Journal on Applied Signal Processing **8** (2003) 791–805
5. Dahlstedt, P.: A MutaSynth in parameter space: interactive composition through evolution. Organised Sound **6**(2) (2001) 121–124
6. Johnson, C.G.: Exploring sound-space with interactive genetic algorithms. Leonardo **36**(1) (2003) 51–54
7. Mandelis, J., Husbands, P.: Genophone: Evolving sounds and integral performance parameter mappings. In: EvoWorkshops. (2003)
8. Takagi, H.: Interactive evolutionary computation: Fusion of the capabilities of EC optimization and human evaluation. Proc. of the IEEE **89**(9) (2001) 1275–1296
9. Bolton, S.: XSynth-DSSI (2005) `http://dssi.sourceforge.net`, last viewed 2 March 2006.
10. McDermott, J., Griffith, N.J.L., O'Neill, M.: Evolutionary Computation Applied to Sound Synthesis. In: The Art of Artificial Evolution. Springer (2006)
11. Beyer, H.G.: The Theory of Evolution Strategies. Springer (2001)
12. McDermott, J., Griffith, N.J.L., O'Neill, M.: Timbral, perceptual, and statistical attributes for synthesized sound. In: Proc. of the International Computer Music Conference. (2006)