Interactive EC Control of Synthesized Timbre

James McDermott jamesmichaelmcdermott@gmail.com
Natural Computing Research & Applications Group,
University College Dublin, Ireland

Michael O'Neill m.oneill@ucd.ie
Natural Computing Research & Applications Group,
University College Dublin, Ireland

Niall J. L. Griffith niall.griffith@ul.ie
Department of Computer Science & Information Systems,
University of Limerick, Ireland

Abstract

Perhaps the biggest limitation of interactive EC is the fitness evaluation bottleneck, caused by slow user evaluation and leading to small populations and user fatigue. In this study these problems are addressed through the proposal of new user interface techniques for interactive EC, which allow faster evaluation of large numbers of individuals and the combination of interactive with noninteractive evaluation. For the first time in the interactive EC literature a set of rigorous usability experiments compares these techniques with existing interactive EC and non-EC interfaces, for the application domain of sound synthesis. The results show that a new user interface for interactive EC improves performance, and further experiments lead to refinement of its design. The experimental protocol shows, again for the first time, that formal usability experiments are useful in the interactive EC setting. Statistically significant results are obtained on clearly-defined performance metrics, and the protocol is general enough to be of potential interest to all interactive EC researchers.

Keywords

Interactive EC, genetic algorithms, sound synthesis, graphical user interfaces, usability experiments.

1 Introduction

Interactive EC is important in many aesthetic, subjective, and multi-objective domains (Takagi, 2001). Perhaps its biggest open problem is the *fitness evaluation bottleneck*: human evaluation of results is slow, so users become fatigued, small population sizes and a limited number of generations must be used, and performance suffers (Takagi, 2001; Biles, 1994; Johanson and Poli, 1998; Llorà et al., 2005). Many papers are published each year attempting to improve performance and on other interactive EC topics, but an important step in the maturing of this young field has not yet been taken: there has been little work using formal usability experiments to measure any performance improvements. Such experiments are certainly difficult, by comparison with purely computational experiments, and perhaps some researchers regard them as impossible. Here, however, a rigorous protocol is used successfully for usability experiments, and

© 2010 by the Massachusetts Institute of Technology

Downloaded from http://direct.mit.edu/evco/article-pdf/18/2/277/1493993/evco.2010.18.2.18205.pdf by guest on 18 June 2021

statistically significant results are obtained on clearly-defined performance metrics. The experimental protocol is general enough to be of potential interest to all interactive EC researchers.

These experiments are used to provide support for the other main contributions of this paper: two new techniques for interactive EC which are intended to improve performance by allowing faster evaluation of large numbers of individuals and the combination of interactive with noninteractive evaluation. These techniques are here tested on the problem of controlling a sound synthesizer, but are general enough to be used elsewhere.

The control of sound synthesis is an interesting application domain for interactive EC methods, for several reasons. Firstly, in this domain the quality of an individual's phenotype—that is, an output sound—is amenable to evaluation both by signal-processing methods (measuring sound similarity) and by users (via their ears). The domain thus straddles the border between interactive and noninteractive EC, and in this paper the latter is used in a role subordinate to the former.

Sound synthesis is also extremely popular: judging by web forum membership (see, e.g., KVR Audio, 2008), it seems reasonable to guess that there are millions of software synthesizer users worldwide. It is hard to think of another application domain in which such a large number of users are faced with the control of nontrivial parametric systems, and this represents, perhaps, an underexploited testing ground for EC ideas.

The problem of controlling a synthesizer genuinely is nontrivial: the numbers of parameters range from a handful to the hundreds, though a few dozen is not atypical. That their names and numerical values tend to be rooted in underlying signal-processing algorithms, rather than reflecting perceptual qualities of the output sounds, only increases the difficulty of control for the (often untrained) users. The parameters may behave nonlinearly and are often epistatic, meaning that the effect of one may depend on the values of others. On the other hand, parameters' behaviour and interactions are not so chaotic and complex as to make EC no better than a random search. There is exploitable structure in the fitness landscape, as is demonstrated later.

1.1 Aims and Methodology

The aim of this work is to design a new interactive EC system leading to performance improvements, and to measure any such improvements in a rigorous way. Such a system must first include new techniques specific to the case of interactive EC. It must also be suited to the particular characteristics of the problem domain: here, interactive EC is applied to the problem of setting sound synthesis parameters. This problem presents different challenges and opportunities from work in, for example, engineering design. Analysis of the new techniques with reference to the problem domain is therefore required (though the new ideas presented here may be applicable to other domains also). Most importantly, the system must be tested in controlled experiments, and refined in light of their results. These requirements dictate the structure of our research, as follows.

Sound synthesis itself is first introduced in Section 2.1. The review of existing work (Sections 2.2–2.2.2) gives an analysis of some of the most important problems in interactive EC, and some previous work on their solutions. This review leads directly to the requirement of new and better interactive EC techniques, and two of these are proposed. Section 3 describes *sweeping*, a user-controlled interpolating operator for

Downloaded from http://direct.mit.edu/evco/article-pdf/18/2/277/1493993/evco.2010.18.2.18205.pdf by guest on 18 June 2021

interacting with and efficiently selecting from an evolutionary population. This is a novel operator for this domain.

Background evolution is the second technique to be proposed, in Section 4. It is a novel type of island-model algorithm, taking advantage of the nature of the problem domain by using both human and machine evaluation of individuals. To run such an algorithm it is necessary to establish that noninteractive evolution can be successful. This question is also addressed in this section.

The bulk of the results are presented in Section 5 and concern a set of formal interactive experiments run to measure the performance of these two techniques. Such experiments, relatively rare in the interactive EC literature, constitute the ultimate test of new interactive techniques. The experimental protocol adopted here can serve as a template for researchers interested in supporting their innovations in interactive EC with experimental data. The purpose of each experiment is to allow the refinement of some methods and the rejection, at least in the scope of this paper, of others. The rationale for the choice of experiments is given more fully in Section 5.

Section 6 contains discussion of the results and the possibilities for using these new techniques in other domains, and finally Section 7 summarises a framework for experiments in interactive EC and gives conclusions.

Downloaded from http://direct.mit.edu/evco/article-pdf/18/2/277/1493993/evco.2010.18.2.18205.pdf by guest on 18 June 2021

2 Background

2.1 The Control of Sound Synthesis

A sound synthesizer is any device or process, whether realised in hardware or software, which produces an audio signal through some form of calculation or algorithm (as opposed to mere reproduction). Modern synthesizers are most commonly implemented as software.

The control of a synthesizer can be roughly divided into two parts. First, notes of varying pitch, volume, and duration can be produced using a controller such as a MIDI keyboard. Secondly, the *timbre* or character of the notes can be controlled by varying the synthesis algorithms themselves and their numerical input parameters. Most synthesizers are *parametric*—that is, the user can vary numerical parameters but cannot add, remove, or recombine components of the synthesis algorithm. Setting the parameters to the right values to give a desired output sound is the problem of interest here.

Most synthesizer users work on controlling timbre interactively via a graphical user interface (GUI) with one controller (such as an on-screen knob, switch, or slider) per parameter. Setting the parameters manually can be a difficult task, because of the sheer number of them, and because they are usually continuous-valued and may be epistatic. In general there is no direct "backward" mapping from a sound to the *patch* (i.e., the set of parameter values) that gives rise to it. The parameter space may be too large to be searched exhaustively, whether manually or through machine methods. In this context, EC is one natural solution.

2.2 Interactive EC

Interactive EC, simply, is EC with a human "inside the loop": usually the user provides either numerical evaluations of fitness, or direct selection of individuals, with other parts of the EC algorithm running as usual. Other types of human involvement in

the algorithm are also possible, but much less common. In most cases, the user is not required to understand the details of the evolutionary algorithm or the underlying problem or representation, and therefore interactive EC is suitable even for untrained users.

Interactive EC is used especially in aesthetic domains in which a computational fitness function is difficult to define, and is also common where the user must make sense of multiple conflicting objectives. It has been used extensively in generative graphics (Dawkins, 1986; Sims, 1991; Todd and Latham, 1992), engineering design (Parmee, 2001), musical composition (Biles, 1994), and many other areas (Takagi, 2001).

2.2.1 EC Sound Synthesis

In particular, interactive EC has been applied to sound synthesis. A typical system is described by Johnson (2003). Here, genomes are floating-point arrays which are translated into sounds by (i.e., serve as input to) a parametric 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 a combination of mutation and crossover. For the population size of 9, each generation therefore requires at least 10 mouse clicks and nine slider movements. This system is perhaps the most direct possible translation of a typical GA into the interactive setting.

A relatively low number of generations is reported to be required for successful evolution—three or four for some convergence, and a few more for more detailed exploration of a subspace. Success is reported both in this respect and in the ability of novice users to learn the capabilities of the synthesizer being controlled.

Several more interesting features are reported as being in development, including a load/save facility; local search search by mutation of a single sound; and user control of per-gene mutation rates.

MutaSynth (Dahlstedt, 2001) is an interactive EC application that can be applied to different synthesis engines via MIDI control, and can also control 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 system uses a population of nine individuals, each represented on screen by *turtle* graphics derived from the parameter values. This innovation provided the user with a potentially useful mnemonic for each individual in the population, and allowed the user to immediately see relationships within and between generations.

Two nonstandard crossover operators, called *insemination* and *morphing*, were introduced on the assumption that the user's intuition would be a guide to the type of combination operator required to produce a desired result from a pair of parents. Another nonstandard operator was *manual mutation*, or direct setting of individual parameter values: parameter values could also be "frozen" manually by the user.

One great strength of this work is a consideration of the multiple synthesizer engines to be placed under interactive EC control, and of the pros and cons of various ways of avoiding typical problems associated with them. For example, the *interdependence* between parameters in some synthesizer models is noted as a problem; a possible solution, parameter mappings that avoid bad combinations of parameter values, is proposed; and the resulting loss of flexibility is noted as a disadvantage. Another solution is to start from good genomes and use only small mutations; but again there is a loss of freedom.

Downloaded from http://direct.mit.edu/evco/article-pdf/18/2/277/1493993/evco.2010.18.2.18205.pdf by guest on 18 June 2021

An interesting insight is the claim that the more universal the synthesizer used, the smaller the proportion of useful sounds that occur in the search space. More general synthesizers, in other words, are much more difficult to control through EC methods, whereas less flexible ones may not even require any extra method of control other than their original parameters.

Genophone (Mandelis and Husbands, 2006) is a complex interactive system in which a data glove, an evolutionary software interface, and a MIDI keyboard and synthesizer are used together. The data glove is used to "perform" with synthesizer parameters via user-variable performance mappings, while the keyboard can be used to perform note data in the usual way. Evolution takes place at the level of synthesis parameters and that of performance parameters: both are represented together in the genotypes, and therefore the user in awarding fitness must take account of both the aesthetic value of the current sound and the practical value of the current performance mappings. The evaluation process therefore is necessarily time-consuming, since the user must (unless the output sound is obviously poor) spend time learning the possibilities of the current mapping.

In terms of EC, the process again involves nonstandard operators, including a (noninteractive) interpolation and a fitness-weighted probabilistic crossover. The user selects individuals interactively, but also specifies a fitness value for each. The mutation and reproduction operators are then applied at the user's discretion. Offspring are awarded preliminary fitness values that depend on those of their parents. The user can choose to "lock" certain synthesizer parameters, limiting the effects of later operations.

The overall process is not intended for target-matching uses, but for exploration. In this context, it is difficult to carry out quantitative experiments. However, qualitative experiments are described, in which users were encouraged to pursue their goals through controlled techniques. Thus, for example, the different crossovers were compared, and the standard one-point crossover was found to be the most predictable and therefore most useful. The explanation offered for this is that interdependent parameters were very often located close together on the genome. The synthesizer in use in *Genophone* was a commercial MIDI synthesizer, in which parameters were to some extent grouped by function, so the explanation is likely true; but the conclusion is stated without quantitative support.

Overall, the system is seen to be a success: users enjoyed using it. The metaphor of variation and inheritance was seen to be easily understood by nontechnical users. This may be in contrast with systems based on alternative metaphors such as the neural network. A related point is made by Gartland-Jones and Copley (2003, p. 45): "GAs map human exploration processes," in the sense that the GA paradigm reflects the creation, exploration, and optimisation methods used in human innovation.

All three of these systems have been reported to be quite successful. They share many common features: one notable one is the *locking* or *freezing* feature present in some form in all three. It is also present in the *Metasurface*, a nonevolutionary interpolating synthesis controller (Bencina, 2005). The apparently independent *convergent evolution* of multiple systems on this feature is evidence that it is particularly appropriate to these parametric systems. One disadvantage to this is a leakage from the evolutionary abstraction, since the user is no longer insulated from low-level details; however, the use of the locking feature is always optional. This means that the systems are capable of being used at different levels of proficiency, from novice to expert.

The interactive EC systems discussed here have been used largely for exploratory, rather than target-matching behaviour. This is certainly a reasonable goal for research,

Downloaded from http://direct.mit.edu/evco/article-pdf/18/2/277/1493993/evco.2010.18.2.18205.pdf by guest on 18 June 2021

particularly since it is a common use-case in real-world synthesizer use. However, it seems natural to ask whether the systems *can* be used to match clearly-specified targets, and if not, why not. One possibility is simply that interactive EC is an inadequate search technique for this problem, given that a desired sound is likely to correspond to a very small proportion of the search space. However, if the aim is to make synthesis control easier for novices, then this use-case must be considered also.

2.2.2 Problems for Interactive EC, and Some Solutions

Takagi (2001) lists the significant problems characteristic of the interactive domain. There is a *fitness evaluation bottleneck* (Biles, 1994)—human evaluation of solutions is naturally far slower (at least for the vast majority of problem domains) than machine evaluation. Not only must the individual solutions be considered (aurally, visually, or otherwise, depending on the domain) and evaluated (generally using keystrokes or mouse gestures), but users often choose to revisit each member of a population multiple times to compare them with each other, before performing the final evaluation. Bad user-interface design can slow a user down, perhaps by requiring too many mouse gestures for each generation. Boredom can arise if a user is asked to evaluate too many similar sounds. Frustration can occur when the system is not *seen* to be learning from and responding to the user's choices. The result of all these problems is that the large populations and numbers of generations, typical of EC in general, are not feasible in interactive EC. As a result, interactive evolutionary algorithms, viewed purely as search algorithms, are at a severe performance disadvantage. Also, the user's goal is usually underdefined and somewhat changeable. This point is sometimes made by characterising interactive EC as searching for a *moving* needle in a haystack. The user may also become progressively more discriminating as evolution proceeds.

Many authors have worked on solutions to some of these problems. One possibility is to somehow model the *implicit fitness function* being applied by the user, so that some individuals can be evaluated automatically and some of the workload removed from the user. Llorà et al. (2005) had some success with this method, though on a toy problem. Johanson and Poli (1998) used a neural network to model the user's implicit fitness function in a GP-style music composition system. Training such a system requires the user to provide many fitness evaluations in a consistent setting (i.e., while the user's goals remain constant). Therefore, this method does not provide a shortcut to fast evolution for later runs in different contexts, with different goals, or by different users. Baluja et al. (1994) avoided this problem by collecting fitness evaluations from many users in advance of a noninteractive run, but this meant that fitness was merely an overall index of general quality, and not specific to a user or relative to a particular goal.

Another problem is specific to the audio domain. Sound is a time-based medium, and there is no easy way to present more than one sound to a user simultaneously. This is in contrast with the graphical domain, where *thumbnails* are routinely used, allowing the user to judge an entire population quickly and focus in on a single individual when necessary.

In this paper, two methods of avoiding the problems of interactive EC are proposed. One, *background evolution*, is a new type of island model. It combines interactive and noninteractive evolution: this can be seen as forming a type of fitness model analogous to systems discussed above. It is specific to the problem instance and does not require the gathering of training data in advance. The other, *sweeping*, can be seen as analogous to the method of thumbnailing: it allows efficient evaluation of a population, a focussing-in

Downloaded from http://direct.mit.edu/evco/article-pdf/18/2/277/1493993/evco.2010.18.2.18205.pdf by guest on 18 June 2021

of attention where necessary, and fast selection. In the case of sound synthesis, it does so by taking fuller advantage of the properties of the synthesizer and the auditory system. Both background evolution and sweeping address the problem of the fitness evaluation bottleneck. They are described in the following two sections.

New interactive EC methods which are claimed to improve performance require experimental support. All three of the interactive EC synthesis systems described above (Johnson, 2003; Dahlstedt, 2001; Mandelis and Husbands, 2006) are reported to be successful and useful for *exploration* of the sound-space. However, little research has been reported on attempting to find specific target sounds—an important use-case. Experiments are also lacking which compare these methods with other possible methods of control, including nonevolutionary methods. The same is true of the fitness-modelling techniques described above (Llorà et al., 2005; Johanson and Poli, 1998; Baluja et al., 1994). This paper attempts to address this lack through controlled usability experiments comparing the innovations with standard interactive EC and non-EC methods of control. These experiments are described in Section 5.

Downloaded from http://direct.mit.edu/evco/article-pdf/18/2/277/1493993/evco.2010.18.2.18205.pdf by guest on 18 June 2021

3 Sweeping

One possible solution to the problem of small populations and poor performance in interactive EC is to find methods that allow larger populations to be evaluated quickly by the user. One such method is proposed here: *sweeping*. This is a user-controlled interpolation between members of a population in which the results are observable in real time. A single controller affects multiple parameters, effecting a gradual transformation between individuals, and the user controls the movement with the benefit of direct feedback. Eventually, the user finds one or more of the best individuals available in the interpolation and selects them directly, without awarding numerical fitness values. Sweeping thus avoids the need for explicit fitness and the traditional selection operator.

Sweeping where one controller corresponds to one synthesis parameter is ubiquitous in nonevolutionary synthesis control, and has even given rise to musical motifs characteristic of synthesized music, such as the *resonant filter sweep*. Authors and software implementers including Bencina (2005) have also used mappings in which a single controller corresponds to multiple synthesis parameters: these trade individual control of parameters for user-interface simplicity. The idea has not previously been applied to sound synthesis EC.

3.1 Benefits of Sweeping

Although it is generally necessary to listen closely to an entire sound (perhaps multiple times) in order to judge it as good, it is often possible to judge that a sound is bad on the basis of a superficial or incomplete listening. One major benefit of sweeping and the direct selection method is that they allow the user to take advantage of this asymmetry. By moving the interpolation controller quickly, the user achieves a superficial impression of many individuals, and can choose to ignore poor-quality areas and focus on better ones. The overall effect is to allow a more superficial method of control which, it is hoped, is also more efficient.

Although the sweeping method allows the user to access only a very small portion of the search space at any one time, it is emphasised that this portion is larger than that accessible through the more typical discrete population. The number of individuals

accessible through the sweeping technique is much larger than suggested by the nominal population size (which will be just three in the first implementation).

The parameters of a synthesizer can be regarded as defining (via their Cartesian product) a continuous, bounded, multi-dimensional *timbre space*. Research in the field of auditory perception (Grey, 1975; Wessel, 1999, and others), has supported the idea that listeners are capable of a type of geometric interpretation of such a space. Listeners asked the question "what sound is to *C* as *A* is to *B*?" are able to respond, and responses conform to a parallelogram law on the timbre space (Ehresman and Wessel, 1978; McAdams and Cunibile, 1992). Grey (1975) suggests that interpolations in a timbre space can be correctly interpreted by listeners. Interpreting the space geometrically informs navigation within it.

It seems likely that listeners can use these abilities to imagine the results of movements in an interpolation, with reference to known sounds, and to guess where a desired sound is likely to be found. This can lead to improved performance (in sounds evaluated per unit time, and in usability) over systems that do not allow the user to apply this intuition in choosing what user interface gestures to make. At least three results of this geometric intuition are hypothesised. A user listening to a poor quality sound will know that a large jump is required to escape the poor quality area: a great number of poor quality sounds can be avoided in this way, preventing user boredom and frustration. Similarly, when the current sound is close to that desired, the user intuitively applies small changes. This combination of coarse and fine control is a key advantage of the sweeping user interface. Finally, the user may sometimes judge not only that the current sound is undesirable, but that the most recent movement in the interpolation was *in the wrong direction*, and that further similar movements are unlikely to be successful. This ability allows the user, in many cases, to change direction and so avoid culs-de-sac.

3.2 Requirements for Effective Sweeping

The hypotheses mentioned above are based on the assumption that the mapping from interpolation to phenotype is well behaved in certain ways. A small movement in the controller should result in *a small change to the phenotype*; a further controller movement in the same direction should result in a further change *in the same (phenotypic) direction*.

These requirements are fulfilled in the case of sound synthesis. They equate to requiring continuity and monotonicity in the mapping from controller to *perceived timbre* (Bencina, 2005; Goudeseune, 2003). This map is composed of two parts: the map from controller to parameter space, and the map from parameter space to timbre space. It is easy to see that the former (defined in Section 3.4 and Figure 1) is continuous and monotonic. The latter component is much more complicated and depends on the synthesizer in use. Most synthesizers are sufficiently well behaved that interpolation is quite easy to use, in practise: the details are outside the scope of this paper but are described elsewhere (McDermott, 2008).

A more practical consideration is that the mapping be computable in real time, that is, fast enough for the user to take advantage of immediate feedback from each change. Sound synthesizers are generally written to be controllable via real-time parameter manipulation and therefore fulfill this requirement also.

3.3 EC View

The sweeping technique is incorporated into an EC algorithm as follows. The standard selection method—awarding numerical fitness values to individuals as the basis for

Downloaded from http://direct.mit.edu/evco/article-pdf/18/2/277/1493993/evco.2010.18.2.18205.pdf by guest on 18 June 2021

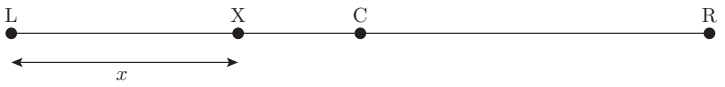


Figure 1: The sweeping operator works by linear interpolation. When the slider is at a proportion x of the distance from L to C, the output patch $X = L + x(C - L)$. The corresponding equation is used when the slider is between C and R.

algorithmic (e.g., roulette wheel) selection—is replaced with a direct selection, performed by the user, among the (relatively few) individuals of the population and the (more numerous) individuals available via interpolation among them. After selection, individuals may go directly into a new population, or they may be processed by crossover and mutation operators. Only the former possibility is used in this paper. Since the best individuals are always retained, the technique can be regarded as *elitist*: in the absence of save/load facilities in the GUIs, it seems necessary that the user should never find that the preferred individuals are lost between generations.

It will be apparent that an evolutionary algorithm that allows the user to intervene directly at the genotype level, observing the phenotypic effects in real time, and that allows the propagation of these genotypic changes to the next generation, is neither a true GA, strictly defined, nor a close analogy to real-world genetic evolution. In fact, the sweeping method may be regarded as Lamarckian, or as an example of the Baldwin effect (Anderson, 1997).

The evolutionary mechanism behind the sweeping interface can also be compared with an evolutionary strategy or ES (Beyer, 2001). Key characteristics of the ES include small populations; an interpolating, multiple-parent recombination operator; and the side-by-side evolution of both the candidate population and a "strategy set" that determines evolutionary parameters. The latter has not been explored in the present work, but the other features have.

3.4 Implementation

The two dimensions of a computer screen make a two-dimensional implementation of sweeping most natural. However, a one-dimensional implementation has also been implemented, with the aim of running the simplest possible experiments. This is described first.

3.4.1 One-Dimensional Sweeping

In one dimension, interpolation is simple. Given two patches L and C (which are simply the floating-point array genotypes of two individuals), the output patch X (i.e., the genotype resulting from this operator) depends on the location of the controller, expressed as a proportion x of the distance between them:

$$X = L + x(C - L)$$

As stated earlier, this mapping is continuous and monotonic. The reason for the choice of patch names is that in the GUI implementation, two such interpolations are juxtaposed, as in Figure 1. The nominal population size is thus increased to three (L, C, and R). The controller can be moved freely between the two interpolations (the mapping is

Downloaded from http://direct.mit.edu/evco/article-pdf/18/2/277/1493993/evco.2010.18.2.18205.pdf by guest on 18 June 2021

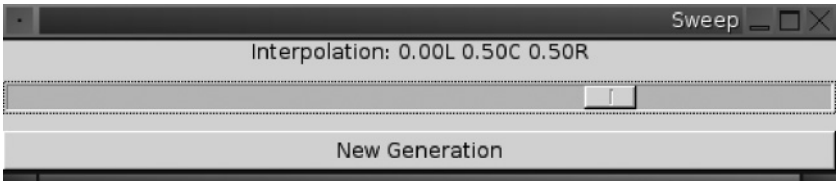


Figure 2: Sweep GUI: the patches L, C, and R are at the left, centre, and right of the slider, and their proportions in the current output patch X are indicated.

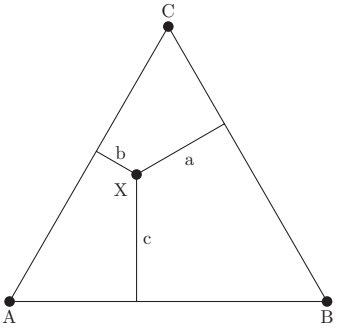


Figure 3: Interpolation using a triangulation scheme. The output patch $X = \frac{a}{a+b+c}A + \frac{b}{a+b+c}B + \frac{c}{a+b+c}C$, where a, b, and c are defined as the lengths of the perpendiculars at X. The coefficients of A, B, and C always sum to 1 and therefore the output is a weighted sum of these patches.

continuous but nondifferentiable at the centre point C). When the user clicks "New Generation" (see Figure 2), the current patch X replaces C—in other words, a single elite point is selected and becomes centred in the next generation. In the first implementation, new patches at L and R are generated randomly, though other possibilities are also discussed later.

The simple GUI presented here can be extended as follows. Multiple independent controllers can be created, each containing three individuals as before. The current output patch is that defined by whichever controller was last touched by the user. The "New Generation" button iterates the population for each of the controllers.

3.4.2 Two-Dimensional Sweeping

In two dimensions, an interpolation can be defined to make it possible to produce any desired linear combination of three patches arranged in a triangle, but not any larger number (howsoever arranged). The interpolation used here is quite simple, as defined in Figure 3. The coefficients of the three patches always sum to 1 and therefore the output is a weighted sum of these patches.

Again, multiple interpolations of the basic form are juxtaposed, to allow centering of an elite individual and a greater choice within each generation. Six triangular interpolations are juxtaposed to give a hexagonal arrangement as in Figure 4. This is comparable to the triangulation method used by Van Nort et al. (2004). In this version of the GUI,

Downloaded from http://direct.mit.edu/evco/article-pdf/18/2/277/1493993/evco.2010.18.2.18205.pdf by guest on 18 June 2021

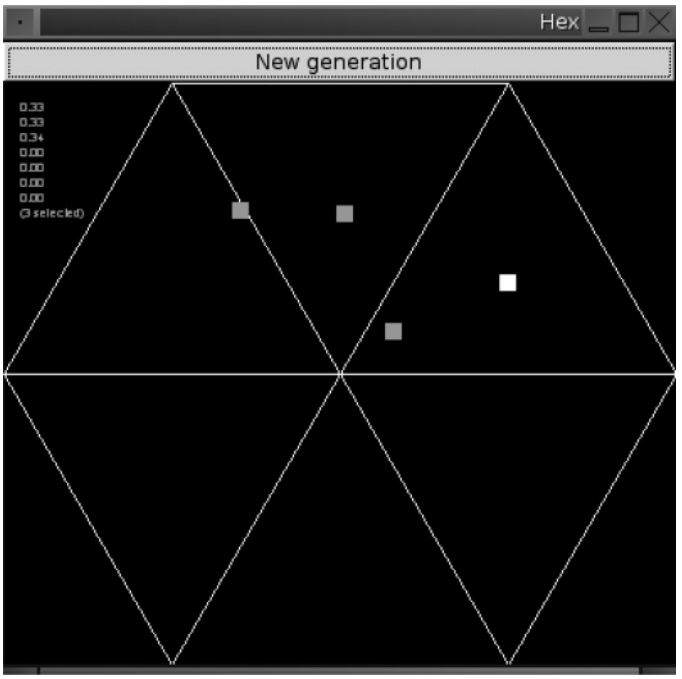


Figure 4: Hex GUI. The numbers at the top left indicate the current proportions (a, b, and c in Figure 3) associated with the vertices. Vertices are ordered starting at the centre and proceeding to the rightmost and then anticlockwise. The proportions (0.33, 0.33, 0.34, 0, 0, 0, 0) indicate the position (indicated by the cursor, in white) in the centre of the first (top right) triangle. Several individuals, marked in grey, have been selected for the next generation.

multiple patches can be directly selected (by right-clicking), to be passed through to the next generation. When the current generation has been sufficiently explored, the user clicks "New Generation" and saved patches are redeployed to the vertices (with the first saved patch going to the centre, and the remaining ones going clockwise from the rightmost). Any vertices remaining unfilled are filled with randomly-generated patches.

4 Background Evolution

Another possible solution to the problem of slow evaluation in interactive EC is to transfer some of the evaluation workload from the user to the computer. There are several methods for achieving this, as discussed in Section 2.2.2, but the one proposed here is a novel type of island-model algorithm: *background evolution*.

The idea of background evolution is as follows: the user interacts through a GUI with an interactive EC system, performing fitness evaluation or direct selection, while in the background a noninteractive EC process runs, occasionally pushing good individuals to be incorporated into the interactive EC (foreground) population. This scheme combines

Downloaded from http://direct.mit.edu/evco/article-pdf/18/2/277/1493993/evco.2010.18.2.18205.pdf by guest on 18 June 2021

the complementary strengths of human and computational evaluation and is possible only in domains where the quality of an individual is amenable to both.

In the case of sound synthesis, background evolution works by first loading a target sound recording, assumed to have some of the qualities desired for the output sound, to function as a target. A noninteractive evolution process is started in which fitness is calculated as similarity to the target (to be defined later). A second process starts a GUI with which the user interacts, driving evolution as usual for interactive EC. The two processes are independent except for occasional copying of good individuals from the noninteractive background evolution to the interactive evolution, to be incorporated into the user's GUI for evaluation.

Downloaded from http://direct.mit.edu/evco/article-pdf/18/2/277/1493993/evco.2010.18.2.18205.pdf by guest on 18 June 2021

4.1 EC View

Since background evolution maintains separate populations with some migration between them, it can be regarded as a type of multiple-deme or island model GA, as discussed by Martin et al. (1997). One way in which the background evolution scheme differs from typical island model GAs is that here migration is one way: from the background process to the foreground. The alternative—that some individuals selected by the user also pass into the background population—is a promising possibility, but is left for future work.

4.2 Computational Measures of Sound Similarity

The key problem in using noninteractive EC to match target sounds is defining a computational measure of sound similarity. Typically this is done by analysis of the sound signals using DSP (digital signal processing) methods. A measure based on the comparison of sounds' spectra (via the discrete Fourier transform or DFT) is the dominant method in the EC synthesis literature (Wun and Horner, 2005; Horner, 2003; Horner et al., 1993; Mitchell and Sullivan, 2005). It may be possible to define measures that are better correlated with human perception of sound similarity, and this may lead to improved results. Different measures have different effects on the fitness landscape and problem difficulty, and this also affects the quality of results.

Previous work (McDermott, 2008; McDermott et al., 2007) has offered a comparison of four different computational measures of sound similarity:

DFT. The spectral comparison measure mentioned above.

Pointwise. A time domain sample-by-sample comparison.

Attributes. A method based on extracting 40 semantic attributes, such as attack time and spectral centroid, from sounds and comparing the attributes after scaling and normalisation.

Composite. A measure based on the average of the above three.

Experiments using these measures as fitness functions suggested that none was unequivocally better than the others, when judged in terms of (computational) fitness achieved. However, when listeners judged the results of evolutions, the time domain comparison was found to have performed worst, and the other three were roughly equivalent.

Table 1: Measures of problem difficulty summarised.

Difficulty measure	Pointwise	DFT	Attribute	Composite
FDC (higher is easier)	low	lowest	higher	highest
Epistasis (lower is easier)	low	low	high	high
Monotonicity (higher is easier)	lowest	medium	higher	highest

Another set of experiments analysed the difficulty of the fitness landscapes induced by the above measures. Three measures of landscape difficulty were used: *fitness distance correlation* or FDC (Jones, 1995), *epistasis* (Beasley et al., 1993), and *monotonicity* (Naudts and Kallel, 2000). The results are summarised in Table 1: it shows that different measures tend to induce different types of difficulty in the fitness landscape. The details can be found elsewhere (McDermott, 2008).

Finally, experimental data were gathered on the correlation between listener's judgements of sound distance and the values returned by the four computational measures. The attribute distance function was shown to be the best correlated of the distance measures. Overall, considering this result, the results on landscape difficulty and previous results on evolutionary success (McDermott, 2008; McDermott et al., 2007), the attribute measure was chosen for sole consideration in later experiments. Note also that since the background evolution must run in parallel with a real-time sound synthesis task, it is desirable to avoid a CPU-heavy fitness function: the attribute function is moderate in CPU use, compared to the composite function which requires calculation of all the others.

4.3 Noninteractive Experimental Results

In order to run background evolution, it is necessary to show that noninteractive EC can succeed in matching targets. A series of experiments was run to demonstrate this and to investigate the best EC algorithms and algorithm parameters for use in noninteractive target-matching. The synthesizer used in all following experiments was a slightly modified version of the Xsynth synthesizer (Bolton, 2005). It has 29 parameters, of which four are integer-valued (encoded as floating-point) and the others true floating-point parameters. This makes the search space considerably larger than that used in much existing EC synthesis work. The genome contained one floating-point number in $[0, 1]$ per parameter. This was mapped to the corresponding parameter's range linearly or logarithmically—some parameters are marked by the synthesizer as logarithmic, meaning that controllers should perform such a mapping to achieve appropriately fine control of the extremities of the parameter's range.

4.3.1 Genetic Algorithm Versus Random Search

The first experiment was carried out to compare the performance of a random search with that of a GA, with the hypothesis that the GA would perform better. The GA was a steady-state GA over 100 generations with 100 individuals in the population. The replacement proportion was 0.5, one-point crossover had a probability of 0.5, and Gaussian mutation had a per-gene probability of 0.1. Selection was by the roulette wheel algorithm. This amounts to a fairly typical floating-point GA. The random search algorithm randomly generated a large number of individuals, returning the best. In order to allow fair comparison between two search algorithms, it is only necessary that

Downloaded from http://direct.mit.edu/evco/article-pdf/18/2/277/1493993/evco.2010.18.2.18205.pdf by guest on 18 June 2021

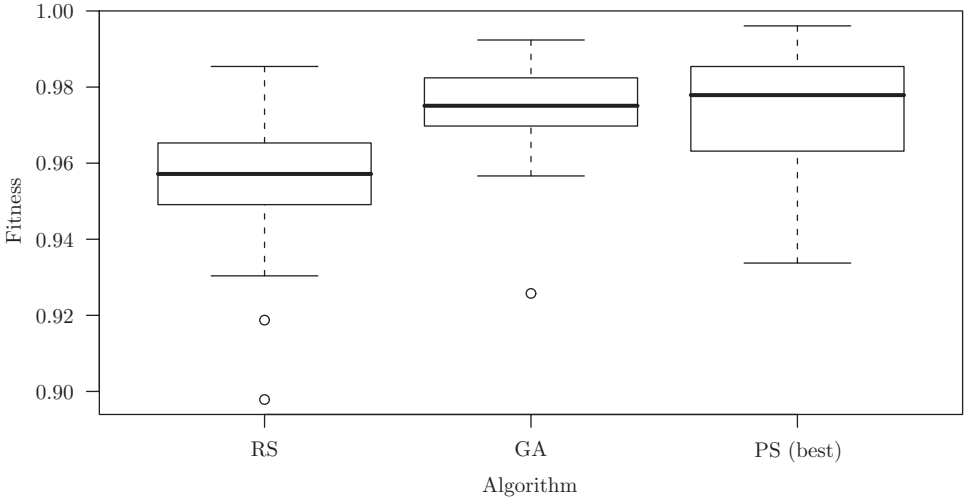


Figure 5: Random search versus GA. The GA achieves better (higher) fitness values. The best result from a particle swarm optimisation—discussed later—is included for comparison.

they perform equal numbers of fitness evaluations, that is, that they process the same numbers of individuals. A steady-state GA with parameters as above will evaluate 5,000 individuals, so the same number was allowed for the random search.

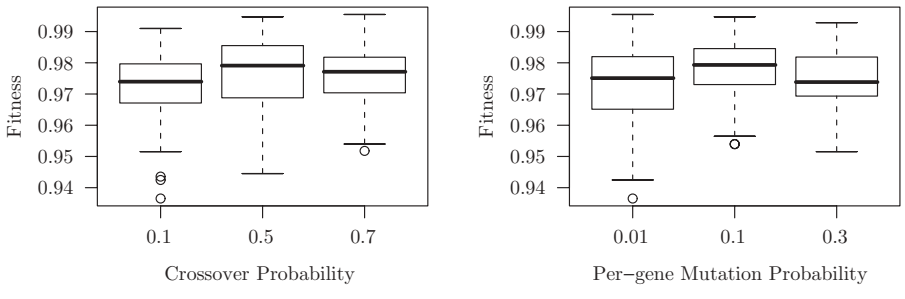
The attribute distance function was used to define fitness as $f = 1/(1 + d)$, where f is fitness and d is the attribute distance between the candidate and target sounds. It was used to drive evolution 30 times, once for each of 30 randomly-generated target sounds. The results in the form of best final fitness values are shown in Figure 5: the GA achieves better (higher) fitness values in each case than the random search. The t-tests among the datasets show that for each of the four distance functions, the GA out-performed the random search ($p < .01$). This is an encouraging result, showing that the problem has at least some exploitable structure.

4.3.2 Genetic Algorithm Parameter Tuning

Next, an experiment was run to test the performance of a GA with various values for the crossover and mutation probabilities. The setup was similar to the previous experiment, using the same number of runs, the same target sounds and the same population size and number of generations. It was hypothesized that both the mutation and crossover probabilities would affect the performance of the GA, as measured by the attribute distance function. The results are shown in Figure 6.

According to a two-way ANOVA, there were significant effects due to the crossover probability ($p < .05$, $F = 4.3$), the mutation probability ($p < .01$, $F = 6.0$), and the interaction between the two ($p < .05$, $F = 3.1$). A Tukey honest significant differences test was run to compare the setups pairwise, and showed that among the crossover probabilities, 0.5 was significantly better than 0.1 (the other differences being insignificant at the $p < .05$ level). Among mutation probabilities, 0.1 was shown to be better than either of 0.01 and 0.3. Among the combined factor levels, the (crossover, mutation)

Downloaded from http://direct.mit.edu/evco/article-pdf/18/2/277/1493993/evco.2010.18.2.18205.pdf by guest on 18 June 2021



(a) Crossover (b) Mutation

Figure 6: Performance analysed by crossover and mutation probabilities. The typical values of 0.5 for crossover and 0.1 for mutation are the best.

combinations (0.5, 0.3), (0.1, 0.1), (0.7, 0.01), and (0.5, 0.1) were shown to perform *better* than (0.1, 0.01). This worst-performing combination featured the lowest probabilities of crossover and mutation, leading to the interpretation that insufficient introduction of diversity and mixing of genes took place in the populations under those conditions. It is noted that the best-performing levels of the factors—0.5 for crossover and 0.1 for mutation—are the most typical of continuous GAs. These values are used in the remainder of the study.

Several other experiments on noninteractive evolution were also run (McDermott et al., 2007; McDermott, 2008). In every case, the aim was to improve performance in order to run more successful background evolution. Among the possibilities examined were:

- Alternative fitness functions, as described above.
- Alternative search algorithms, including the comparison of particle swarm and genetic algorithms (the best particle swarm results obtained are shown in Figure 5, for comparison).
- *Increasingly discriminating fitness functions*, where an attribute-based fitness function was composed initially of only a few attributes, but of an increasing number as evolution proceeded (the aim being to allow the population to evolve good building blocks on easy subproblems and later work to combine them).
- The weighting of components in the attribute fitness function.

However, in no case were clear-cut improvements found over the best GA setup described above. Therefore the GA with the attribute fitness function was chosen for the practical implementation of the background evolution idea, described next.

4.4 Implementation of Background Evolution

The following implementation was used: individuals are pushed from the background evolution to a stack whenever a new individual exceeds the previous best fitness achieved in the background population; individuals are popped (last-in, first-out) from this stack to the GUI whenever the user requests a new generation in the foreground

Downloaded from http://direct.mit.edu/evco/article-pdf/18/2/277/1493993/evco.2010.18.2.18205.pdf by guest on 18 June 2021

evolution. If the user requests a new generation when the stack is empty (either because no individuals have yet been pushed onto it, or because all existing individuals have been popped off and no new ones yet pushed on), then a random individual is generated instead.

Pilot experiments and the results given above on automatic target-matching inform this design in several ways. The population size (50) in the background evolution is chosen to give a regular stream of improvements (too large a population would cause long gaps between new generations) and the best overall performance within the time frame envisaged for trials of the target-matching experiments described below (3–6 min).

A buffer of good individuals is necessary for cases when the user requests several new generations (and thus individuals from the background process) in quick succession. A last-in, first-out buffer (i.e., a stack) is the best choice, since the most recent best individual found by the background process is the most likely to be of use to the user.

In the next section, the sweeping and background evolution techniques described above are tested in a series of formal usability experiments.

5 GUI Experiments

A series of three usability experiments was run to test the performance of the sweeping and background evolution techniques proposed in previous sections, and to inform their further design.

These experiments are intended to allow the refinement of some methods and the rejection, at least in the scope of this paper, of others. However, it is not practical to attempt experimental investigation of every variation on the interactive methods. This is because interactive experiments are very time-consuming, relative to purely computational ones, but moreover because they tend to yield less clear-cut results. The GUIs tested here are deliberately kept simple: implementation details (such as synthesizer-specific ones) and convenience features (such as loading and saving) are not treated here. This approach is chosen because it allows the statement and testing of clear hypotheses concerning minimal GUIs, it imposes as little cognitive load as possible on users, and it prevents extranaeous issues from interfering with the core results. Each experiment therefore tests the simplest possible conditions. Thus, the experimental process is not intended to lead necessarily to the best possible design of an interactive EC synthesis system, but to explore fundamental design choices.

5.1 Measuring Success

A key question is how performance can be measured. Attention is restricted to a single type of task—target-matching—which does not represent the totality of possible synthesizer use-cases, but is at least somewhat amenable to objective methods. Multiple subjects perform multiple tasks in a factorial experimental design, so that the effect of the GUIs on performance can be isolated. For each target-matching task, there are two main measures of success, as follows. After the task, the user is asked to give an estimate (from a scale of 1–7) of the similarity between the final, best sound achieved and the target. This estimate is purely subjective, and subjects are likely to differ in the criteria on which their estimates are based. However, there is no reason to suppose that this subjectivity biases results for or against any of the GUIs being tested. The second measure of success is the elapsed time. Several computational measures of sound similarity were also calculated between each result and the corresponding target sound; however, these are *not* regarded as capable of overruling human perception of sound similarity.

Downloaded from http://direct.mit.edu/evco/article-pdf/18/2/277/1493993/evco.2010.18.2.18205.pdf by guest on 18 June 2021

5.2 Experimental Setup

The three experiments were very similar in their setup. For each, 20 subjects volunteered. Roughly equal numbers of male and female subjects volunteered, their ages ranged from early 20s to mid-40s, and their self-reported abilities and knowledge in music and synthesizers varied from none to expert level. There was some overlap in subjects between the three experiments. Some of the subjects were students or graduates of a course in music technology, and some were personally known to the authors. Each subject received a nominal payment of €5.

Two target sounds were chosen from among the preprogrammed sounds created by the synthesizer author. They were thus relatively desirable or "musical," in comparison with the majority of available sounds. A third sound, used only in the final experiment, was created from scratch in order to test the ability of GUIs to find sounds with a particular set of perceptual features.

The first two experiments compared four GUIs, while there were three in the final experiment. Since there were two target sounds for the first experiments, and three for the final one, each subject performed a total of eight or nine target matching tasks in total. The order of presentation of the GUIs was varied between subjects, so that effects of learning or fatigue did not bias the results for or against any GUI. Subjects were advised to take as long as they wished on each task, but warned that a fixed number of tasks was to be completed. A period of 3–6 min per task was suggested, chosen as being sufficient for a typical synthesizer user to achieve a reasonable first approximation to a desired sound under normal circumstances.

5.3 Three Experiments

In addition to the target-matching tasks, subjects performed short preliminary listening tests, for example, choosing which pair of three sounds was the most similar, and rating the similarity of pairs of sounds. These experiments gathered data useful in the study of the search space (McDermott, 2008) and had the effect of "warming subjects up" to the fine discrimination and careful listening necessary in the main tasks.

5.3.1 Non-EC, Interactive EC, and Interactive EC Innovations

In this first experiment, four GUIs were tested; the first two functioned as controls against which to measure the success of the two new interactive EC techniques. The simplest possible implementations of these techniques were used, in order to avoid ambiguity in the interpretation of results.

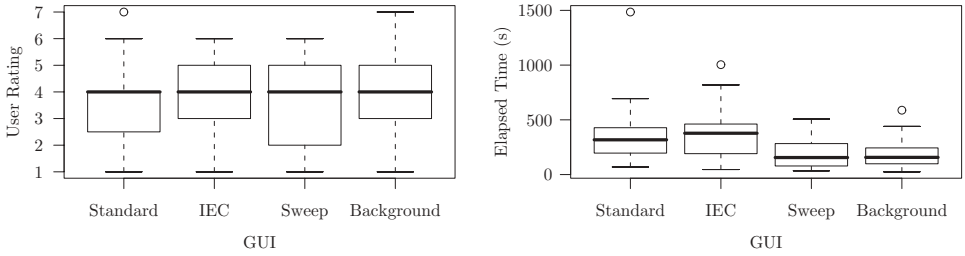
Slider. A typical non-EC GUI in which each synthesis parameter was controlled independently by a slider.

IGA. A typical interactive EC GUI (shown in Figure 7), similar to that of Johnson (2003), with a population of eight, numerical fitness values in a scale of 1–7, one-point crossover with probability 1.0, and Gaussian mutation of deviation width 0.75 and probability 0.3.

Sweep. A simple sweeping GUI, with random generation of end points at each new generation.

Downloaded from http://direct.mit.edu/evco/article-pdf/18/2/277/1493993/evco.2010.18.2.18205.pdf by guest on 18 June 2021

Figure 7: Typical interactive GA GUI.



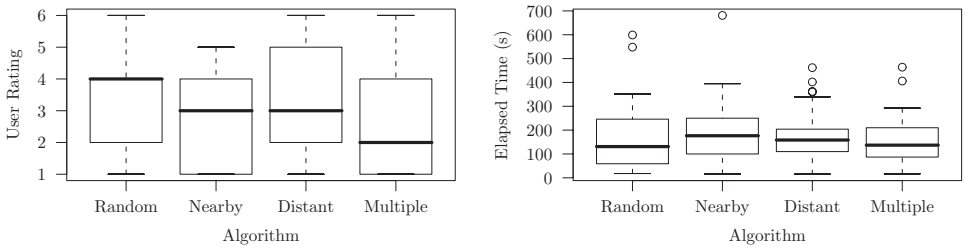
(a) User rating (b) Elapsed time

Figure 8: Results analysed by GUI. For user rating, higher is better; for elapsed time, lower is better.

Sweep with Background Evolution. A sweeping GUI where one of the two end points at each generation was copied from a background evolution, as explained in Section 4.4.

The results are given in Figure 8. They were analysed using two-way repeated measure ANOVAs. There are no statistically significant differences in user rating between GUIs ($F = 0.38$; $p > .1$). However, there are significant differences by target ($F = 34.8$; $p < .001$) and by GUI against target ($F = 3.14$; $p < .05$). The user rating results separated out by target sound suggest that the GUI against target sound interaction effect is that the sweep GUIs are more suitable to target 0 than target 1. This 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 sound 1), while the sliders GUI 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. The addition of background evolution (i.e., the contrast between the third and fourth GUIs) was not found to increase performance—this is despite the first-hand impression of the

Downloaded from http://direct.mit.edu/evco/article-pdf/18/2/277/1493993/evco.2010.18.2.18205.pdf by guest on 18 June 2021



(a) User Rating (b) Elapsed Time

Figure 9: Results of the second experiment, analysed by algorithm.

Downloaded from http://direct.mit.edu/evco/article-pdf/18/2/277/1493993/evco.2010.18.2.18205.pdf by guest on 18 June 2021

first author that the background evolution technique did make better "raw material" available for evolution.

Figure 8(b) suggests that there was a strong effect of GUI on elapsed time, and the ANOVA confirms this ($F = 12.1$; $p < .001$). In this case, the post-test pairwise t-tests show that subjects used the two sweep GUIs for significantly shorter times than the other two GUIs. This is a positive result for the sweep GUIs, since achieving the same quality of match in a shorter time benefits both the serious and the casual user. The addition of background evolution (i.e., the contrast between the third and fourth GUIs) did not have a significant effect on elapsed time.

The subjects' self-reported ability in music and synthesis (encoded as three levels—beginner, intermediate, and advanced) was shown not to have any effect on performance at the $p < .05$ level. This is a somewhat surprising result, since one motivation for interactive EC techniques is that they hide complexity and are therefore more suitable for novices. It is possible that the more expert subjects' better abilities were partly cancelled out by their stricter standards of similarity. In any case, subjects' questionnaires did reveal that of the nine 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 [non-EC] GUI didn't have that."

There was a lack of evidence supporting the background evolution method. Although it is argued later (in Section 6.2) that the idea is not without merit, the evidence suggested that the idea not be pursued further in the course of the current research. The sweeping idea, on the other hand, achieved promising results, and the next stage of the research focussed on refining it.

5.3.2 End Point Generation Methods

Perhaps the aspect of the sweeping operator most amenable to variation is the method of generating end points. It was hypothesized that more sophisticated end point generation algorithms might lead to better performance than the random generation used in the previous experiment. Four different algorithms were implemented in otherwise identical sweeping GUIs:

Random. Random generation (as before).

Nearby. Random generation with a constraint that end points be relatively similar (at a parameter distance of 0.06, as discussed below) to the centre point.

Distant. Random generation with a constraint that end points be dissimilar (at a distance of 0.45).

Multiple. Random generation with the constraints that they be somewhat dissimilar from each other and from individuals of previous generations (at a distance of more than 0.06 in each case).

The similarity constraints were defined numerically using a measure of distance in the synthesizer's parameter space. For two points, each parameter value was mapped (linearly or logarithmically, as appropriate) to the range [0, 1], and then the difference between corresponding values was taken and the results averaged, giving an overall measure of distance, itself in the range [0, 1]. The distance 0.06, used above, was chosen as just large enough to enforce discriminability between end points (McDermott, 2008). The value 0.45 was chosen as being close to the maximum distance (0.5) constructible in general from a given input point.

The motivating hypotheses for the choice of end point algorithms were that increasing diversity, and enforcing distinctions between end points and previous generations, would improve performance. Thus, this is an attempt to discover the correct balance between exploration and exploitation. We note that the "distant" method not only tends to increase population diversity, but also means that a given movement in the interpolation always corresponds to the same distance in parameter space—that is, it leads to diversity and consistency.

There were significant differences ($p < .01$, $F = 4.3$) in user rating by algorithm. Post-test pairwise t-tests showed that the "distant" algorithm performed better than the "nearby" and "multiple" algorithms at the $p < .05$ confidence level, after a Bonferroni correction of a factor of 20 to allow for multiple statistical tests on the same data. The differences between the "random" algorithm and all of the others were *not* found to be significant. Neither the algorithm nor the target affected the elapsed time results (elapsed time by algorithm: $p > .1$, $F = 0.44$; elapsed time by target: $p > .1$, $F = 3.7$). The effect of the interaction between target and algorithm was not significant either ($p > .1$, $F = 1.8$).

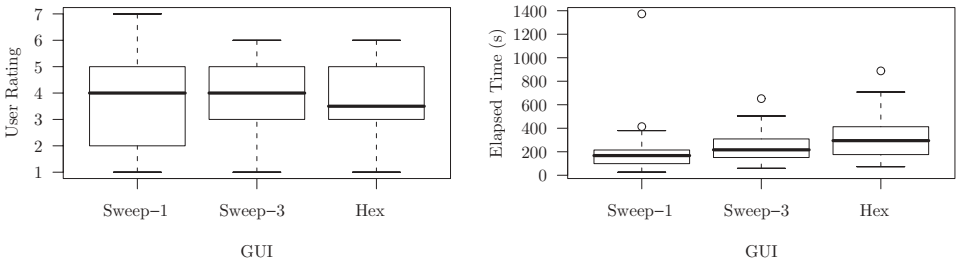
The fact that statistical significance is found in some results is evidence that performance improvements can be detected where they exist, and gives strong support for the methodology adopted in this experiment.

The idea of multiple constraints between the end points and previous points, and between the end points themselves, was shown to decrease performance. It is possible that this occurs because the enforcement of novelty with respect to previous generations sometimes moved the population away from promising areas being explored by the user. The difference in performance between the "nearby" and "distant" algorithms shows that increased diversity improves performance. The distance constraint enforced in the "distant" algorithm (0.45 in parameter distance), cannot be increased greatly for further gains, since 0.5 is the largest possible distance achievable in general from a given input point. Nevertheless, the relative success of the "distant" algorithm suggested that it be used in all further experiments.

5.3.3 Extended Sweeping Methods

As stated earlier, a one-dimensional sweeping method was used initially for the sake of simple, controlled experiments. However, a two-dimensional method is the most natural for a typical screen and mouse interface. Again, the simplest possible extension

Downloaded from http://direct.mit.edu/evco/article-pdf/18/2/277/1493993/evco.2010.18.2.18205.pdf by guest on 18 June 2021



(a) User rating (b) Elapsed time

Figure 10: Results analysed by GUI for the third experiment.

Downloaded from http://direct.mit.edu/evco/article-pdf/18/2/277/1493993/evco.2010.18.2.18205.pdf by guest on 18 June 2021

was chosen. The hexagonal system, described in Section 3.4.2, consists of juxtaposed triangular interpolations, just as the system used in the earlier experiments consists of juxtaposed linear interpolations. The two were here compared directly. To try to disentangle two effects—the number of dimensions, and the nominal population size—a third interface was tested which still used only one-dimensional interpolation, but increased the number of individuals available per generation.

In both of the previous experiments, target sound 0 was found to be much easier to match. This result was confirmed by a noninteractive method of sampling the search space and calculating the mean similarity to the target sounds, using DSP-based measures. A third target sound was defined in order to gather more data on this phenomenon. To keep the overall experiment duration relatively short, the number of GUIs tested was reduced to three.

Sweep-1. A sweeping GUI using the "distant" end point generation algorithm, as above, functioning as a control.

Sweep-3. A GUI using three independent sweeping elements, as described in Section 3.4.1.

Hex. A two-dimensional sweeping GUI, formed of six triangular interpolation elements in the shape of a hexagon, as described in Section 3.4.2.

Thus, the experiment tested two alternative methods of extending the sweeping method—an increased population for more parallel search, and a two-dimensional approach—comparing both to a control.

Figure 10 gives the results for each of the three GUIs. They suggest the trend that the more complex the GUI (from Sweep-1 to Sweep-3 to Hex), the longer subjects spent with it, though without achieving better results. This impression is confirmed by statistical analysis, which shows that the GUI did not have a significant effect on user rating but did on elapsed time: subjects spent longer using the Hex GUI than either Sweep-1 or Sweep-3 ($p < .01$ after a Bonferroni correction with a factor of 10 to allow for the ANOVA and several pairwise t-tests). Note that one trial using GUI 0 was recorded as having continued for almost 1,400 s, and is marked as an outlier in Figure 10(b). It is likely that this trial lasted so long due to an error of some sort, perhaps the subject choosing to take a break during a trial rather than between trials. When this subject's

data is *temporarily* excluded from the dataset, the difference in elapsed time between Sweep-1 and Sweep-3 is also shown to be significant at the $p < .01$ level, after Bonferroni adjustment.

Of the eight subjects who expressed a preference in their post-test questionnaires, seven preferred the Hex GUI—though one of these claimed that while the Hex GUI was the most interesting and enjoyable to use, the Sweep-3 GUI was the most successful. Two subjects said that the Sweep-1 GUI was too simple.

6 Discussion

It is to be expected that the subjective nature of the domain and the noise inherent to user experiments will not always lead to strong, clear-cut conclusions. However, a tentative suggestion can be made as to the "best" system to emerge from these experiments. It is clear that the sweeping technique is feasible and attractive to users. An algorithm enforcing diversity and consistency of behaviour through selection of end points seems to be useful and well worth incorporating. On the other hand, the technique of background evolution was not supported by this work, and is perhaps not worth implementing in a real-world system—at least not in this domain, or not without further research. The two-dimensional sweeping method appeals strongly to users, but the one-dimensional method produced equivalent results faster. In practice, it may be worthwhile to provide both, and this is the approach taken by the first author in code for a real-world music composition program (Ritter and Holtzhausen, 2008).

6.1 Effective Sweeping

The novel GUI based on sweeping was shown to be competitive with typical interactive EC and non-EC GUIs, and overall subjects said that they preferred it. Some methods of generating the end points for interpolation were investigated, and a method giving a type of consistency of behaviour and increased population diversity was shown to give improved performance over two methods lacking these characteristics. Two possible extended sweeping methods were investigated, and subjects found that two-dimensional sweeping was viable and interesting.

Sections 3.1 and 3.2 set out the requirements for sweeping to be effective in a given domain. It must be possible for the map from representation to solution (i.e., from genotype to phenotype) to be computed in real time, and it must be possible for the user to experience and judge the solution in real time also. Sound synthesis fulfills these requirements, at least for short sounds, whereas entire musical compositions would take too long to be evaluated.

The mapping from interpolation to phenotype must be perceived to be well-behaved, that is, monotonic and smooth, and the user's geometric intuition in the search space must not be violated. Otherwise, the interpolation degenerates to random search. That these conditions are fulfilled in other application domains appears plausible—for example in the case of graphics it is possible to imagine the result of interpolating between individuals (e.g., Hart, 2007). However, a new argument, taking account of the details of each domain and of human perception of its phenotypes and phenotypic changes, must be made in each case.

It was argued in Section 3.1 that sweeping allows fast, superficial evaluation of many bad individuals, saving the user time, but still allows focussing-in on good areas of the interpolation. This was facilitated by the asymmetry that exists in the evaluation

Downloaded from http://direct.mit.edu/evco/article-pdf/18/2/277/1493993/evco.2010.18.2.18205.pdf by guest on 18 June 2021

of sounds: it is often possible to judge a bad sound as being bad on the basis of a superficial or incomplete listening, while judging a sound as good may require more careful listening. This asymmetry (which is largely absent in noninteractive EC) also exists in other aesthetic domains, including graphics.

The sweeping technique appears potentially useful in many domains where solutions are amenable to either graphical or instantaneous sonic representation. Areas such as generative graphics, image processing, industrial design, and hearing aid fitting are among those listed by Takagi (2001, p. 1277) and potentially amenable to the sweeping technique. We can add sound synthesis to this list, and suggest new areas such as data visualisation and sonification. Our own research continues into the use of sweeping in other aesthetic domains, and into the underlying representations most suitable for this operator. The sweeping idea, implementation details, and experiments therefore constitute a contribution to the overall state of interactive EC knowledge.

Downloaded from http://direct.mit.edu/evco/article-pdf/18/2/277/1493993/evco.2010.18.2.18205.pdf by guest on 18 June 2021

6.2 Background Evolution in Other Domains

The method of background evolution was not shown to improve performance in the first experiment, and was not investigated further in the later ones. As originally presented, background evolution is likely to be only occasionally applicable in real-world uses: it can only be helpful when the user possesses a recording that demonstrates some of the qualities desired for the output sound. This may seem unrealistic, in that a user might prefer to simply use the recording, rather than looking for a resynthesised version of it. However, synthesis gives flexibility in pitch, duration, and expression (plus the possibility of further timbral manipulation), which may not be available through playback techniques. Many synthesizer users never attempt any type of resynthesis of an existing file, and so this type of usage is rare; but it may be argued that this is partly due to the difficulty of manual resynthesis, and that the technique might become more popular given a good automatic tool.

The ability to add knowledge to an evolution by setting a target may be useful as a new workflow in synthesis and other types of design software. A user of graphical EC might load a picture with a desired palette as the background target, even though it has none of the structural features that the user will select for in the foreground. The idea is that migration and crossover will later combine a good palette and good structures.

Another possibility comes from the domain of 3D design. 3D modelling software typically includes a library of prebuilt designs, and many architects and designers are familiar with a workflow in which these designs are taken as starting points in the creative process. However, background evolution allows a designer to specify a target that is not included in the library and does not exist in a representation editable within the software. A photograph or a hand-drawn sketch could be made the target of evolution, so that a successful evolution results in an editable (and further evolvable) representation of an otherwise inaccessible design. Similar techniques may be applicable to other aesthetic, subjective, and multi-objective domains.

Background evolution is not limited to cases where a desired object already exists. Instead, it is possible to define the computational fitness function without reference to any target. It is quite common to define fitness functions in aesthetic domains in terms of abstract measures such as symmetry, size, density, and variation (Dahlstedt, 2007; O'Reilly and Hemberg, 2007). Using these measures, the user can direct background search while retaining interactive control of the foreground population. Computational fitness functions might also be defined in terms of automated feature recognition in

graphics, or structural analysis in the case of architectural design (Machwe and Parmee, 2007).

Fitness functions based on measures rather than explicit targets might also provide a solution to a problem which arose in some interactive experiments. Some subjects reported that occasionally too many sounds in the populations were very quiet or even silent. This suggests first that the volume parameter, being easy to understand, could be removed from EC control and placed under direct control of the user. A more interesting idea is to use background evolution or a similar idea to avoid this problem. The system could perform a simple local search or filtering to remove from the population candidate sounds—here, silence and near-silence—which are obviously bad, before the user encounters them. This idea would help to avoid user fatigue, a key problem in interactive EC, and it is surprisingly general: even purely aesthetic domains, apparently inaccessible to computational fitness functions and measures of similarity, may allow some filtering of obviously-bad individuals. This point is related to that of the asymmetry in evaluation time between good and bad solutions, made in the previous section.

7 Conclusions

This paper makes two main contributions to the field of interactive EC—new techniques intended to improve performance, and for the first time in the literature, a rigorous experimental protocol intended to measure any such improvements.

Previous work on interactive EC and sound synthesis has been reviewed, identifying the main problems and lack of formal usability experiments. The idea of *sweeping* has been introduced with support from the literature on human audio perception and investigation of the genotype-phenotype mapping. It allows efficient interaction with and selection from the population. The idea of *background evolution*, which combines interactive with noninteractive EC, is informed by experiments on and analysis of the noninteractive target-matching problem. These new techniques have been tested against other interactive EC and non-EC methods of control, and further refined, in three sets of formal usability experiments. The sweeping technique was found to improve performance and to appeal to users, and the experimental approach found to be successful in terms of delivering significant results.

Several possibilities have been suggested for future work within this domain, and in extending the new techniques to other interactive domains. It is emphasised that arguments specific to sound synthesis can be reformulated for other applications. In every case the argument requires some domain-specific knowledge, but might follow the framework set out in this paper.

- A continuous representation is used.
- The behaviour of the genotype-phenotype mapping is investigated with regard to interpolation usability, and it is noted that it can be computed in real time.
- The literature on human perception in the domain is used to argue in favour of an interpolating approach.
- Noninteractive EC using computational measures of similarity on the domain is demonstrated to be viable, as a precursor to the implementation of background evolution.

Downloaded from http://direct.mit.edu/evco/article-pdf/18/2/277/1493993/evco.2010.18.2.18205.pdf by guest on 18 June 2021

- Possible sources of targets for noninteractive EC in the domain are suggested.
- Performance measures including elapsed time and user-perceived similarity to targets are defined.
- A factorial experimental design is carried out, with both novice and expert users as appropriate, and statistical analysis of results.

A similar approach to experiments may be of benefit in investigating the performance of other interactive EC innovations, whether domain specific or not.

It will be noted that the GUIs presented here are deliberately simple: more sophisticated implementations of (non-EC) interpolation and of interactive EC synthesis have been presented elsewhere, as described in Section 2.2. The motivation for this simplicity is to allow the statement and testing of clear hypotheses concerning minimal GUIs, to impose as little cognitive load on the user as possible, and to prevent extranaeous issues from interfering with the core results.

This approach has been vindicated: despite the level of noise and uncertainty inherent to an interactive and subjective domain, statistically significant differences in performance (as measured by user rating and elapsed time) were detected in several cases. This is the first time that a rigorous experimental protocol has been used successfully in this domain. It is to be hoped that similar methods will be applied and results obtained in other applications of interactive EC in the future.

Acknowledgments

The first author was supported under the Embark scheme (grant number RS/2003/68) by the Irish Research Council for Science, Engineering and Technology. Many thanks go to Dr. Jean Saunders and Brian Sullivan of the UL Statistics Consulting Unit and Dr. Fred Cummins of UCD for advice on statistics and experimental design, and to all our volunteers for experiments. Thanks also to the anonymous reviewers.

References

Anderson, R. W. (1997). The Baldwin effect. In T. Bäck, D. B. Fogel, and Z. Michalewicz (Eds.), *Handbook of evolutionary computation*, Chap. C3.4 (pp. 1–7). Oxford, UK: IOP Publishing Ltd. and Oxford University Press.

Baluja, S., Pomerleau, D., and Jochem, T. (1994). Towards automated artificial evolution for computer-generated images. *Connection Science*, 6(2–3):325–354.

Beasley, D., Bull, D. R., and Martin, R. R. (1993). An overview of genetic algorithms: Part 2, research topics. *University Computing*, 15(4):170–181.

Bencina, R. (2005). The Metasurface: Applying natural neighbour interpolation to two-to-many mapping. In *Proceedings of NIME 2005*, pp. 101–104.

Beyer, H.-G. (2001). *The theory of evolution strategies*. Berlin: Springer.

Biles, J. A. (1994). GenJam: A genetic algorithm for generating jazz solos. In *Proceedings of the 1994 International Computer Music Conference*, pp. 131–137. Danish Institute of Electroacoustic Music, Denmark. International Computer Music Association.

Bolton, S. (2005). XSynth-DSSI. Retrieved March 2, 2006, from http://dssi.sourceforge.net/.

Downloaded from http://direct.mit.edu/evco/article-pdf/18/2/277/1493993/evco.2010.18.2.18205.pdf by guest on 18 June 2021

Dahlstedt, P. (2001). A MutaSynth in parameter space: Interactive composition through evolution. *Organised Sound*, 6(2):121–124.

Dahlstedt, P. (2007). Autonomous evolution of complete piano pieces and performances. In *Proceedings of Music AL Workshop*.

Dawkins, R. (1986). *The blind watchmaker*. Harlow, UK: Longman Scientific and Technical.

Ehresman, D., and Wessel, D. (1978). Perception of timbral analogies. Technical Report 13/78, IRCAM, Paris.

Gartland-Jones, A., and Copley, P. (2003). The suitability of genetic algorithms for musical composition. *Contemporary Music Review*, 22(3):43–55.

Goudeseune, C. (2003). Interpolated mappings for musical instruments. *Organised Sound*, 7(2):85–96.

Grey, J. M. (1975). An exploration of musical timbre. PhD thesis, CCRMA, Department of Music, Stanford University.

Hart, D. A. (2007). Toward greater artistic control for interactive evolution of images and animation. In M. Giacobini (Ed.), *Applications of evolutionary computing*, vol. 4448 of *LNCS*, (pp. 527–536). Berlin: Springer.

Horner, A. (2003). Auto-programmable FM and wavetable synthesizers. *Contemporary Music Review*, 22(3):21–29.

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

Johanson, B., and Poli, R. (1998). GP-Music: An interactive genetic programming system. In J. R. Koza, W. Banzhaf, K. Chellapilla, K. Deb, M. Dorigo, D. B. Fogel, M. H. Garzon, D. E. Goldberg, H. Iba, and R. Riolo (Eds.), *Proceedings of the Third Annual Conference on Genetic Programming*, pp. 181–186.

Johnson, C. G. (2003). Exploring sound-space with interactive genetic algorithms. *Leonardo*, 36(1):51–54.

Jones, T. (1995). Evolutionary algorithms, fitness landscapes and search. PhD thesis, University of New Mexico, Albuquerque.

KVR Audio. (2008). KVR Audio forum member list. Retrieved May 20, 2008, from http://www.kvraudio.com/forum/memberlist.php?mode=joined&order=ASC&start=153420.

Llorà, X., Sastry, K., Goldberg, D. E., Gupta, A., and Lakshmi, L. (2005). Combating user fatigue in iGAs: Partial ordering, support vector machines, and synthetic fitness. In *GECCO*, pp. 1363–1370.

Machwe, A. T., and Parmee, I. C. (2007). Towards an interactive, generative design system: Integrating a "build and evolve" approach with machine learning for complex freeform design. In M. Giacobini (Ed.), *Applications of evolutionary computing*, vol. 4448 of *LNCS* (pp. 449–458). Berlin: Springer.

Mandelis, J., and Husbands, P. (2006). Genophone: Evolving sounds and integral performance parameter mappings. *International Journal on Artificial Intelligence Tools*, 15(4):599–622.

Martin, W. N., Lienig, J., and Cohoon, J. P. (1997). Island (migration) models: Evolutionary algorithms based on punctuated equilibria. In T. Bäck, D. B. Fogel, and Z. Michalewicz (Eds.), *Handbook of evolutionary computation*, Chap. C6.3 (pp. 1–16). Oxford, UK: IOP Publishing Ltd. and Oxford University Press.

Downloaded from http://direct.mit.edu/evco/article-pdf/18/2/277/1493993/evco.2010.18.2.18205.pdf by guest on 18 June 2021

McAdams, S., and Cunibile, J.-C. (1992). Perception of timbral analogies. *Philosophical Transactions of the Royal Society*, 336(Series B):383–389.

McDermott, J. (2008). Evolutionary computation applied to the control of sound synthesis. PhD thesis, University of Limerick, Ireland.

McDermott, J., Griffith, N. J. L., and O'Neill, M. (2007). Evolutionary computation applied to sound synthesis. In J. Romero and P. Machado (Eds.), *The Art of artificial evolution: A handbook on evolutionary art and music* (pp. 81–101). Berlin: Springer.

Mitchell, T. J., and Sullivan, J. C. W. (2005). Frequency modulation tone matching using a fuzzy clustering evolution strategy. In *Proceedings of the 118th Convention of the Audio Engineering Society*.

Naudts, B., and Kallel, L. (2000). A comparison of predictive measures of problem difficulty in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 4(1):1–15.

O'Reilly, U.-M., and Hemberg, M. (2007). Integrating generative growth and evolutionary computation for form exploration. *Genetic Programming and Evolvable Machines*, 8(2):163–186.

Parmee, I. C. (2001). *Evolutionary and adaptive computing in engineering design*. Berlin: Springer.

Ritter, L., and Holtzhausen, P. (2008). Aldrin modular sequencer. Retrieved October 27, 2008, from http://www.bitbucket.org/paniq/aldrin/.

Sims, K. (1991). Artificial evolution for computer graphics. In *SIGGRAPH '91: Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques*, 319–328.

Takagi, H. (2001). Interactive evolutionary computation: Fusion of the capabilities of EC optimization and human evaluation. *Proceedings of the IEEE*, 89(9):1275–1296.

Todd, S., and Latham, W. (1992). *Evolutionary art and computers*. San Diego, CA: Academic Press.

Van Nort, D., Wanderley, M. M., and Depalle, P. (2004). On the choice of mappings based on geometric properties. In *Proceedings of the 2004 Conference on New Interfaces for Musical Expression*, pp. 87–91.

Wessel, D. L. (1999). Timbre space as a musical control structure. Technical Report 12/78, IRCAM. Originally published in 1978.

Wun, S., and Horner, A. (2005). A comparison between local search and genetic algorithm methods for wavetable matching. *Journal of the Audio Engineering Society*, 53(4):314–325.

Downloaded from http://direct.mit.edu/evco/article-pdf/18/2/277/1493993/evco.2010.18.2.18205.pdf by guest on 18 June 2021