Eugene: A Generic interactive Genetic Algorithm Controller

C.James-Reynolds[1] and E.Currie[2]

Abstract This paper outlines the development of an open source generic hardware-based interactive Genetic Algorithm controller (Eugene) and explores contexts in which it may be deployed. The system was first applied to the generation of synthetic sound using MIDI and a simple analogue synthesiser with 27 continuous controller values. It was then applied in the area of image evaluation using an image enhancer program with 7 continuous controller values. The system was evaluated by experimental observation of users attempting various tasks with different success criteria. This led to the identification of issues, some of which were specific to, and others divorced from the application domain. These are discussed together with areas for improvement.

1 Introduction

This paper describes the development of an embedded interactive Genetic algorithm controller (Eugene) and its application. We begin by outlining the nature of genetic algorithms and interactive genetic algorithms. We then describe the features of Eugene and finally some evaluation of the system and its interaction with users. We conclude with some discussion of the outcomes and suggestions for future work.

2 Genetic Algorithms

Genetic algorithms (GAs) [1] are examples of evolutionary algorithms used to implement search and optimisation strategies, which loosely parallel Darwinian evolutionary theory. Some GA approaches try to accurately mirror the corresponding biological process [1]. Others have added new processes to assist the user in either reaching an optimal outcome more quickly, or to provide them with the opportunity to explore the solution space more thoroughly.

The GA process starts with a population of candidate solutions or individuals (chromosomes), randomly chosen from the solution space.

[1] School of Science and Technology, Middlesex University, London NW4 4BT
C.James-Reynolds@mdx.ac.uk

[2] School of Science and Technology, Middlesex University, London NW4 4BT
E.Currie@mdx.ac.uk

A fitness function is then used to select those individuals whose characteristics are closest to the desired outcome; these become the parents of the next generation. Each member of the next generation "inherits" variables from each of two randomly chosen parents.

The outcome is either that the system converges on a solution which scores above an agreed threshold value of the fitness function, or that a pre-set number of generations have passed. A good example of how this works in practice is the work on antenna design by [2].

3 Interactive Genetic Algorithms

Interactive Genetic Algorithms (iGAs) have been with us since [3]. They are interactive in the sense that a user's judgement replaces the fitness function used in a traditional GA [4]. The concept is appropriate where evaluation of candidate solutions requires a subjective approach that is difficult to automate; for example the generation of identikit pictures [5].

[6] classifies iGA application areas as Artistic/Engineering/Edutainment, and most published work describes applying iGAs to these areas. There has been some notable work in the area of artistic applications, which often focuses on the media seen as output from the process. [7] and [8] are clearly audio focussed, whereas [9] and [5] are concerned with image processing and identification. Engineering applications include interface design [10], image retrieval and robotics. Edutainment applications include composition support and games.

Although many iGAs have been used in these areas, there are interesting aspects to be explored. For example, should they be tools for those who wish to explore a solution space, or should they be used when there are clearly defined user goal states? [11] Issues such as the fitness evaluation bottleneck are discussed by [12] who also considers the impact of fatigue.

As [13] points out, there is an assumption that users are consistent in their rating across multiple generations and we do not have information on the impact of user fatigue on consistency. [12] also points out that evaluation is often poorly defined in terms of the goals. A goal state may change as the user is exposed to various individuals during the algorithm run.

4 Eugene

The concept for Eugene was one of a simple open source tool that could be easily adapted to work within a wide range of application areas and for which a simple library could be used to allow exploration of some of the issues above. Issues of particular interest to us were the understanding of goals, consistency, and tradeoffs and the ways in which users interact with an iGA interface.

Eugene was developed using the Arduino Uno. It provides six sliders attached to the analogue inputs, with which, in each successive generation, the user rates the appropriateness of each of six candidate individuals (sounds or pictures). The device has play buttons for each of the six candidates attached to digital i/o pins and an evolve button attached to a digital i/o pin, that generates the next population for consideration. One of three possible mutation values may be chosen through the use of a rotary encoder that uses two digital i/o pins. The system generates MIDI Continuous Controller (CC) data on the Arduino serial output pin which is connected to a USB interface via a standard MIDI DIN socket. All the switches are provided with pull up resistors.

Figure 1 Eugene, showing user interface.

MIDI CC data consists of 3 bytes, where the first identifies which of the possible 16 MIDI channels are used and the type of message, the second determines the CC number (0-127) and the third the CC value (0-127). In most software (and some hardware) synthesisers, any control (for example modulation or resonance) can be mapped to any CC number.

When being used for sound synthesis, the MIDI data modifies the settings of a software synthesiser with two oscillators, envelope generators and a state variable filter built in SynthEdit. When used for image correction, the data is used to control a simple image enhancement program allowing modification of 7 parameters (red, green, blue, brightness, contrast, sharpen and blur) developed in Processing by means of the MidiBus library. This potentially allows Eugene to control any application developed in Processing.

Setting a given slider at zero causes that individual to be replaced by a new random individual (the ‘randomise’ enhancement described above), while setting a slider at its maximum position causes that individual to be retained, unchanged, for the next generation (elitism) [8].

On pressing the evolve button, the choice of parents for the next generation uses a roulette wheel approach, where the probability of a given individual being chosen as a parent is proportional to its relative rating by the user.

5 Evaluation

The evaluation was carried out using MIDI to control two very different applications. The first was a 23-parameter model of an analogue synthesiser designed in SynthEdit and the second was a simple image enhancement program built in Processing.

The initial pilot evaluation was carried out with the synthesiser and six users, all of whom had an interest in music or music technology. The users initially played with the controller by experimenting to find sounds they liked. They were then given a target sound and asked to reproduce the sound with Eugene.

One intended limitation of Eugene is that it only triggers the playback of a single note (middle C; MIDI note 60). This simplifies the selection process.

A similar pilot evaluation was performed with the image enhancer program using two users, to identify any major issues. Users were asked to experiment with the controller and then to enhance an image with poor colour, balance and contrast, which was slightly out of focus.

The iGA functioned effectively as a tool for users to explore both search spaces by experimenting with the controller. When given a specific task, users were often quick to eliminate solutions that sounded or looked different from their target. With the synthesiser, they found that it was possible to get close to the desired sounds, but more difficult to get an exact result. With images, both users felt that the initial population was often far removed from their requirements for image enhancement, with images having extreme effects. The users were using the Randomise feature extensively in order to get what they felt was a good starting population and they thought the task would have been far easier in a simple image editor. They did feel that using the system to explore the possibilities was interesting, but that it was of limited practical use without more features.

From the MIDI data logs for the users of the controller with the synthesiser, it could be seen that in their search for a specific sound, users were using Randomise to discard individuals, despite those individuals having variables that were very close to that of the target sound.

An experiment was then conducted to test observations made with subjects using Eugene. Twenty-five subjects were asked to compare six sounds to a reference sound. The sounds were chosen to provide different amounts of variation in the MIDI CC parameters used to generate them. One sound provided a control and was the same as the reference.

The subjects rated each of the sounds for similarity to the reference sound on a scale of 0 to 10, where 0 indicated little similarity to the reference and 10 meant that the sound was almost identical to the reference. The findings were consistent, other than three outliers that distorted the results a little.

The findings showed that modulation of the oscillators (Vibrato) was a change that resulted in very low similarity to the reference, even where other parameters

were identical. Changes in waveshape (Tone) produced very high similarity to the reference where dynamics (changes in loudness) remained similar, and the substitution of one of the oscillators for white noise produced the results with the greatest variance, where users seemed to either hear the sound through the noise and rated it as highly similar to the reference, or they decided that the noise made the sound so unlike the reference that they rated it with a low similarity value.

In terms of the MIDI CC data used to create the sounds, this showed that changes in one variable (e.g. modulation depth or waveshape) can have a more significant impact on users' decisions than larger changes in other variables.

6 Discussion

When we are working with variables that are not continuous, but discrete such as waveform selection in a synthesiser, an individual candidate may be assigned a low score as a result of a single parameter being out by 1 bit. We note that the perception of 'distance' from the desired target is highly sensitive to changes in such parameters.

For control parameters such as oscillator modulation, a low value would probably be desirable if a musical context is sought. For example, vibrato requires a subtle amount of frequency modulation at a few Hertz. This would not necessarily apply if we were looking for a wider gamut of sounds; for example siren effects.

An important conclusion from the observation of the above two scenarios is that the search space and users' cognitive understanding of that search space do not necessarily match and where this occurs there will be a slower convergence on an acceptable solution.

Given that users sometimes discard candidate solutions where the difference in certain parameters is small, we could modify the encoding of the solution space to better match the users' cognitive understanding of the domain. This would allow users to make better decisions and become a more effective fitness function.

This could be resolved to a certain extent by scaling parameters. For example, using a log curve would allow more of the available range of the parameter to be used for smaller changes. This would also, in the case of the image processing software, produce an initial population more suited to the task.

There is also a good argument for limiting the solution space for certain types of application, for example cleanup of images and creative image processing have different solution spaces that may overlap.

The Eugene controller, whilst successful, has a small initial population size and the number of new children is reduced when we retain individuals or remove them (by re-randomising). This reduced amount of available offspring might limit the effectiveness of the algorithm to converge quickly, although our experiments indicate that it does seem to be effective in helping explore the solution space.

7 Conclusion

We have successfully developed a generic hardware iGA controller (Eugene) and applied it to two different domains. The controller facilitated the exploration of the solution spaces effectively, although the simple image processing application did not have as large a solution space as users would have liked. There were however, lessons to be learned from the use of the controller for solving a given task involving a reference target sound or an "ideal" target image.

Many existing iGA approaches have used additional features to enhance the iGA to make it effective for the application domain; future work will incorporate these in addition to supporting a larger initial population. This will provide a better tool for further work.

References

1. Holland, J. H. *Adaptation in Natural and Artificial Systems.* Ann Arbor: University of Michigan Press. (1975).
2. Hornby, G.S., Globus, A., Linden, D.S., Lohn, J.D., Automated antenna design with evolutionary algorithms, in: AIAA Space. pp. 19–21. (2006).
3. Todd, S., Latham, W. Evolutionary Art and Computers. Academic Press. (1992).
4. Biles, J. Genjam: A genetic algorithm for generating jazz solos. In Proceedings of the International Computer Music Conference. (1994).
5. Frowd, C.D., Hancock, P.J.B., Bruce, V., McIntyre, A.H., Pitchford, M., Atkins, R., Webster, A., Pollard, J., Hunt, B., Price, E., Morgan, S., Stoika, A., Dughila, R., Maftei, S., Sendrea, G., Giving Crime the "evo": Catching Criminals Using EvoFIT Facial Composites. IEEE, pp. 36–43. doi:10.1109/EST.2010.38 (2010).
6. Cho, S.-B.,. Towards creative evolutionary systems with interactive genetic algorithm. Applied Intelligence 16, 129–138. (2002).
7. Biles, J.A., Eign, W.G.,. GenJam Populi: Training an IGA via audience-mediated performance. San Francisco, USA 347–348. (1995).
8. Johnson, C.G., Exploring the sound-space of synthesis algorithms using interactive genetic algorithms, in: Proceedings of the AISB'99 Symposium on Musical Creativity. Society for the Study of Artificial Intelligence and Simulation of Behaviour, pp. 20–27. (1999).
9. Takagi, H., Cho, S.-B., Noda, T., Evaluation of an IGA-based image retrieval system using wavelet coefficients, in: Fuzzy Systems Conference Proceedings. Presented at the Fuzzy Systems Conference Proceedings, 1999. FUZZ-IEEE'99. 1999 IEEE International, IEEE, pp. 1775–1780. (1999).
10. Masui, T., Graphic object layout with interactive genetic algorithms, in: Visual Languages, 1992. Proceedings., 1992 IEEE Workshop on. IEEE, pp. 74–80. (1992).
11. Collomosse, J.P., Supervised Genetic Search for Parameter Selection in Painterly Rendering, EvoWorkshops 2006: 599-610 (2006).
12. McDermott, J., O'Neill, M., Griffith, N.J., Interactive EC control of synthesized timbre. Evolutionary Computation 18, 277–303. (2010).
13. Bauerly, M., Liu, Y., Evaluation and Improvement of Interface Aesthetics with an Interactive Genetic Algorithm. International Journal of Human-Computer Interaction 25, 155–166. doi:10.1080/10447310802629801 (2009).