Genetic Programming with User-Driven Selection: Experiments on the Evolution of Algorithms for Image Enhancement

Riccardo Poli∗ and Stefano Cagnoni†
∗School of Computer Science, The University of Birmingham
Birmingham B15 2TT, UK
R.Poli@cs.bham.ac.uk
Phone: +44-121-414-3739

†Department of Electronic Engineering, University of Florence
Via S. Marta, 3, 50139, Florence, Italy
Cagnoni@asp.die.unifi.it

ABSTRACT

In this paper we present an approach to the interactive development of programs for image enhancement with Genetic Programming (GP) based on pseudo-colour transformations. In our approach the user drives GP by deciding which individual should be the winner in tournament selection. The presence of the user does not only allow running GP without a fitness function but it also transforms GP into a very efficient search procedure capable of producing effective solutions to real-life problems in only hundreds of evaluations. In the paper we also propose a strategy to further reduce user interaction: we record the choices made by the user in interactive runs and we later use them to build a model which can replace him/her in longer runs. Experimental results with interactive GP and with our user-modelling strategy are also reported.

1 Introduction

In order to use Genetic Programming (GP) to solve a problem it is usually necessary to define a scalar fitness function or at least a criterion, inducing a total order, for comparing pairs of solutions. However, in many real-world situations it is very difficult, if not impossible, to formulate mathematically a single reasonable quality criterion. This happens, for example, when the quality assessment is based on multiple criteria which are difficult to integrate into a single scalar measure, when subjective or qualitative criteria have to be used or when a training set of input-output pattern-pairs is not available or cannot be created easily.

A large amount of work has been recently devoted to solving the problem of incorporating multiple objective functions in evolutionary algorithms (EAs), the most frequently studied technique being Pareto optimisation (see for example [Goldberg, 1989, Fonseca and Fleming, 1995, Surry et al., 1995, Langdon, 1995, Greenwood et al., 1996]). Somehow less attention, at least in the engineering domain, has been directed to using EAs in the absence of an algorithmic fitness function.

One way of running EAs when a scalar fitness function is not available is to try and induce one from a corpus of available data using machine learning techniques. This method has been used in [Spector and Alpern, 1995], where a set of real jazz-fragments was used to train a neural network. The network was later used to define a fitness function for a GP-based music generation system: the network played the role of a critic, GP was the musician.

A strategy used more frequently to solve this problem is to introduce the human expert/user in the main loop of an evolutionary algorithm and use him or her in the selection phase. This seems the most natural approach when only qualitative/subjective information (such as that provided by humans whose expertise is difficult to elicit and formalise) is available or when the desired output for the program to be induced is not known at all.

For example, Dawkins [Dawkins, 1987] evolved *biomorphs* (insect-like creatures) with an algorithm similar

to evolution strategies in which the user selected which creature would be used to produce a new generation. In his seminal work [Sims, 1991], Sims used Lisp S-expressions with several different mutation operators to evolve programs which produced images and animations on the basis of a user aesthetic criteria. Das *et al.* [Das et al., 1994] used Sims's ideas to represent and evolve interactively 3–D objects and sounds within a virtual-reality environment with GP (including also a recombination operator). Graf and Banzhaf [Graf and Banzhaf, 1995] have recently applied a similar strategy to breed 2–D images and 3–D solid objects interactively using a tiepoint-based representation within a GA. Biles [Biles, 1994] used a GA to evolve jazz solos on the basis of a numeric fitness value defined, entirely subjectively, by a user.

Most of the work summarised above has been motivated by the desire to produce computer art using EAs. However, Gruau *et al.* [Gruau and Quatramaran, 1996] have recently applied this approach (using fitness values assigned, largely subjectively, by the user) within cellular-encoding-based GP [Gruau, 1994] to solve a relatively hard engineering problem: the evolution of neural networks for robot control.

In this paper we describe an approach to the interactive development of programs for image enhancement with GP which can be considered another attempt to explore the possibilities offered by user-steered EAs in the engineering domain. In our approach the user, by being part of tournament selection, controls the evolution of simple programs which enhance and integrate multiple B/W images (time-varying images, multi-modal medical images, multi-band satellite images, etc.) into a single colour image.

Only a very small number of applications of GP to image analysis problems have been reported in the literature to date. The majority of researchers have concentrated on pattern recognition, classification and localisation [Tackett, 1993, Koza, 1994, Johnson et al., 1994, Andre, 1994, Teller and Veloso, 1995] with only some recent efforts considering image enhancement and segmentation tasks [Harris and Buxton, 1996, Daida et al., 1995, Daida et al., 1996, Poli, 1996a, Poli, 1996b]. This paper should also be considered a further extension of our previous work in the area of GP for image enhancement.

The paper is organised as follows. In Section 2 we describe the basic method for user-driven GP, and some ideas on how to reduce or completely eliminate user interaction. In Section 3 we report on the experimental results obtained with the basic method and on some preliminary results with one technique to avoid prolonged user interaction. In Section 4 we make some final comments on our results.

2 Method

Our method for interactive program evolution is not dissimilar from standard generational GP as described in [Koza, 1992] with the only exception of the selection phase, which is driven by the user. The basic method and some extensions which can help reduce or eliminate the user interaction are described in the following two subsections.

2.1 Basic Method

In our approach, instead of asking the user to assign a numerical fitness to all the individuals in a population or to directly select the ones to be used to create the next generation (the strategy used in most of the papers described in the previous section), we ask the user to influence tournament selection by interactively comparing pairs of solutions and determining the winner. By using a tournament size of 2 individuals we avoid the problems of circular preferences, quite common in human judgements, which might make it impossible to choose the winner of the tournament. Circular preferences can arise, for example, when individual A is preferred to B, B to C and C to A. This strategy avoids imposing the use of total or partial orders on the decisions made by the user.

The application chosen for this study was to extend and automate, using GP, the pseudo-colouring techniques described in [Cagnoni et al., 1991] where multiple gray-scale images were combined into a single pseudo-colour image in which some structures of interest were emphasised: a problem for which no standard technique exists. Therefore, in our experiments, selection was simply performed by visually comparing the colour images associated to the two individuals in the tournament and clicking on the preferred one with the mouse.

To maintain near-realtime performance, a crucial feature when the user is part of an evolutionary algorithm, the terminals and functions used for this application are a subset of those used in our previous work on the evolution of filtering programs for image analysis [Poli, 1996a, Poli, 1996b]. The function set included only the operators {`+, -, *, max, min`} and the terminal set included the ephemeral random constant generator [Koza, 1992, pages 242–243] (which produced random numbers in the range $[-1, 1]$) and as many input variables `g1, g2, ...` as the number of images to be integrated/enhanced, where variable `gi` represents the gray level of a pixel of the `i`-th image of the input sequence.

In order to build an output image integrating and enhancing the input sequence, each program in the population was run for each pixel in the input images. The output of each program was clipped to fit in the range $[0, 255]$. The output image was then displayed using a predefined colourmap. In the experiments described in Section 3 we used a "spectrum" colourmap which spans all the rainbow colours by gradually fading from blue to green to yellow and red, with the addition of black and white at the extremes, as shown in Figure 2(a).[1]

In our experiments we used a population size of 20. Such a small size was suggested only by the need to keep the duration of the runs (and, therefore, of the user interaction) at an

[1] This and other pictures in this paper can be best interpreted in colour. They are available at URL:
`http://www.cs.bham.ac.uk/~rmp/eebic/example4.html`.

acceptable level. This is consistent with what is reported in the work described in the previous section, where population sizes from 4 to 32 were used. For the same reason we never run more than 10 to 15 generations. Again this is consistent with the data reported in the literature.

The other parameters used in our runs were: "full" initialisation method with initial program depth of 4, crossover probability of 0.7 and no mutation.

2.2 Extensions to Reduce User Interaction

When the user is part of an evolutionary algorithm only a very limited number of evaluations can be performed, typically of the order of a few hundreds (unless large communities of users are available, e.g. using the World Wide Web). Therefore, although interactive evolution can be used, as reported also by others, to induce quite interesting solutions for moderately complex problems, we should probably not expect it to scale up very well. The reason for this is that more complex engineering problems will almost certainly require populations with hundreds or thousands of individuals which would make the user interaction become extremely time-consuming and boring.[2]

The only way to address the scalability problem deriving from this "user bottleneck" seems to be the development of techniques that reduce or eliminate user interaction but still somehow follow the criteria applied by the user when evaluating or comparing solutions.

To do this we propose a strategy based on the idea that GP may exploit the user's choices to induce a model which can later replace or assist him or her. This can be achieved by first observing and recording the user's choices together with some parameters describing the most important features of the solutions under selection and by later using these data to induce a program mimicking the user's behaviour.

This approach can be applied in several different ways. For example it is possible to apply it in batch mode, executing one or more runs of an interactive EA to collect enough fitness cases, then apply GP to evolve a model of the user choices, and finally use the model as a fitness function or a comparison procedure for large-scale runs. Alternatively, it is possible to record the choices of the user at run-time (e.g. across a few generations or even within a single generation), and build or modify a user model on-line, by running GP every now and then (e.g. at the end of each generation), to maintain the model in close agreement with the user. The model could be used to process the majority of the individuals in the population. Many variations between these fully off-line and fully on-line modes seem possible.

In the preliminary experiments reported in Section 3.3 we used the batch modelling strategy. As the choices of the user were based on the output image produced by each program (phenotype) rather then the structure and size of the program itself (genotype), we decided to extract and record some statistical phenotypical features. In particular, given the relative simplicity of this application and the exploratory nature of this work, we decided to simply use the mean and variance of the output image produced by each program. The values of such parameters for the two images undergoing tournament selection along with the choice made by the user were recorded and used as fitness cases for user modelling. The use of phenotypical features should by no means be intended as the only strategy to gather information on the user's decision criteria: genotypical features (like, for example, the frequency of each function and terminal or the size and shapes of the programs) play, implicitly, a very important role as well.

[2] Although our runs required less than one hour to complete, interactive sessions lasting for up to two days have been described in the literature when the user needs to evaluate complex individuals (see for example [Gruau and Quatramaran, 1996]).

3 Experimental Results

In our experiments we used an implementation of GP developed by the first author in Pop-11, an AI language with garbage collection and incremental compilation widely used in the UK. The experiments are described in the next three subsections.

3.1 Interactive Evolution of Programs for MRI Image Enhancement

In a first set of experiments we have used a pair of Magnetic Resonance (MR) images of the same section of the brain of a patient affected by multiple sclerosis, a disease characterised by the presence of hyper-intense plaques (see Figure 1). The images were acquired using a spin-echo sequence with the following parameters: echo time T_E=50ms and repetition time T_R=2s for the first image, and T_E=100ms and T_R=2s for the second one [Webb, 1988]. Approximately speaking, each image is a map of a different physical parameter: the first image, which we will call PD, strongly depends on proton density, while the second, which we will call RT, on relaxation time T_2. The relaxation time T_2 is the time constant that regulates the decay of the transversal (with respect to the direction of the main static magnetic field) magnetization component induced by a RF pulse that alters the orientation of the magnetic spins of protons. The terminals `g1` and `g2` represent gray levels of pixels in PD and RT, respectively.

Given the very high reproducibility of MR imaging and the minimum inter- and intra-patient variability of the physical characteristics of brain tissues, the use of only one image pair in these experiments does not affect the generality of the solutions found by GP.

In a first set of runs, we tried to evolve programs which emphasised the brain with respect to the ventricle, the white x-shaped structure visible in the middle of RT (see Figure 1), using the basic interactive method described in the previous section. After a few unsuccessful warm-up runs, in the second generation of a run a program with the right kind of behaviour

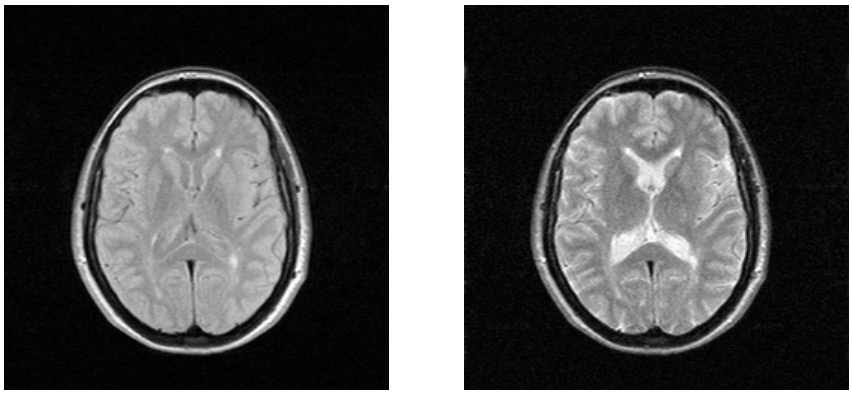

Figure 1: Original MR images used in our experiments: PD (left) and RT (right).

emerged, which we kept refining until generation 5. The refined program is the following:

```
(max (- (* (- -0.070726 g2) (min 0.785222 g1))
        (+ (max -0.315188 0.503258)
           (- -0.650528 g2)))
     (+ (* (- g1 g2) (min g1 g2))
        (- (+ g1 0.404921) (min g1 g1))))
```

which, once simplified, corresponds to the function:

$$\max((-0.070726 - g2)\min(g1, 0.785222) + 0.147270 + g2, (-g2 + g1)\min(g1, g2) + 0.404921).$$

As clarified by the plot of this function shown in Figure 2(b) (after clipping) and by the output image in Figure 2(c) this program mostly behaves like a linear discriminator which zeroes all the pixels which appear very bright in RT (e.g. those belonging to the ventricle) and sets all the non-background pixels to very high values.

In another successful run, we obtained the following program

```
(min (- (min (max 0.123771 g2) (+ g1 g1))
        (+ (- g1 g1) (* 0.751092 -0.131142)))
     (- (max (max -0.136205 g1) (+ g1 g1))
        (- (* g1 g1) (* g1 g1))))
```

which corresponds to the function:

$$\min(\min(2.0g1, \max(g2, 0.123771)) + 0.09849970706, \max(g1, -0.136205, 2.0g1)).$$

The plot of this function is shown in Figure 2(d) while the program output is reported in Figure 2(e). The program uses quite a different strategy to produce images in which the ventricle and the cerebro-spinal fluid are shown in red and the rest of the brain in green. In essence, with minimum approximation, the output image is obtained by taking the minimum between `g2` (the gray level of the pixels of RT) and `2*g1` (the gray level of the pixels of a contrast-enhanced version of PD).

In a second set of runs we tried to evolve programs which would emphasise the multiple-sclerosis plaques of the patient with respect to normal brain tissues. In this case we succeeded in evolving the desired programs at the first attempt. Already in the first generation two or three different programs with the right kind of behaviour were present. When we attempted to refine them in the following generations we realised that every now and then we had to choose between programs which, for us, were basically of the same quality even though they produced quite different outputs. Instinctively we resorted to choosing one program or the other approximately the same number of times. In a sense unexpectedly, during the ten generations of this run GP was able to evolve two phenotypically different lines of solutions at the same time. One of the corresponding genotypes is the program

```
(+ 0.595356 (- (- (- g1 g2) (- g2 g2))
               (- (- g2 g1)
                  (+ (- g2 g1) g2))))
```

which corresponds to the function

$$0.595356 + g1$$

plotted in Figure 2(f), the other is the program

```
(+ (max (* (+ g1 -0.52594)
           (min 0.202965 0.414198))
        (min (+ -0.98135 g1)
             (- 0.825299 g1)))
   (- (- (- g1 g2) (- g2 g2))
      (- (- g2 g1) (+ (- g2 g1) g2))))
```

which corresponds to the function

$$\max(0.202965g1 - 0.1067474121, \min(-0.98135 + g1, 0.825299 - g1)) + g1.$$

As shown by the plot in Figure 2(h), the two genotypes represent two very similar functions: the first is basically PD, while the second is PD multiplied by 1.2. However, due to clipping and to the factor 1.2, the phenotypes shown in Figures 2(g) and (i), are visually quite different, though semantically equivalent.

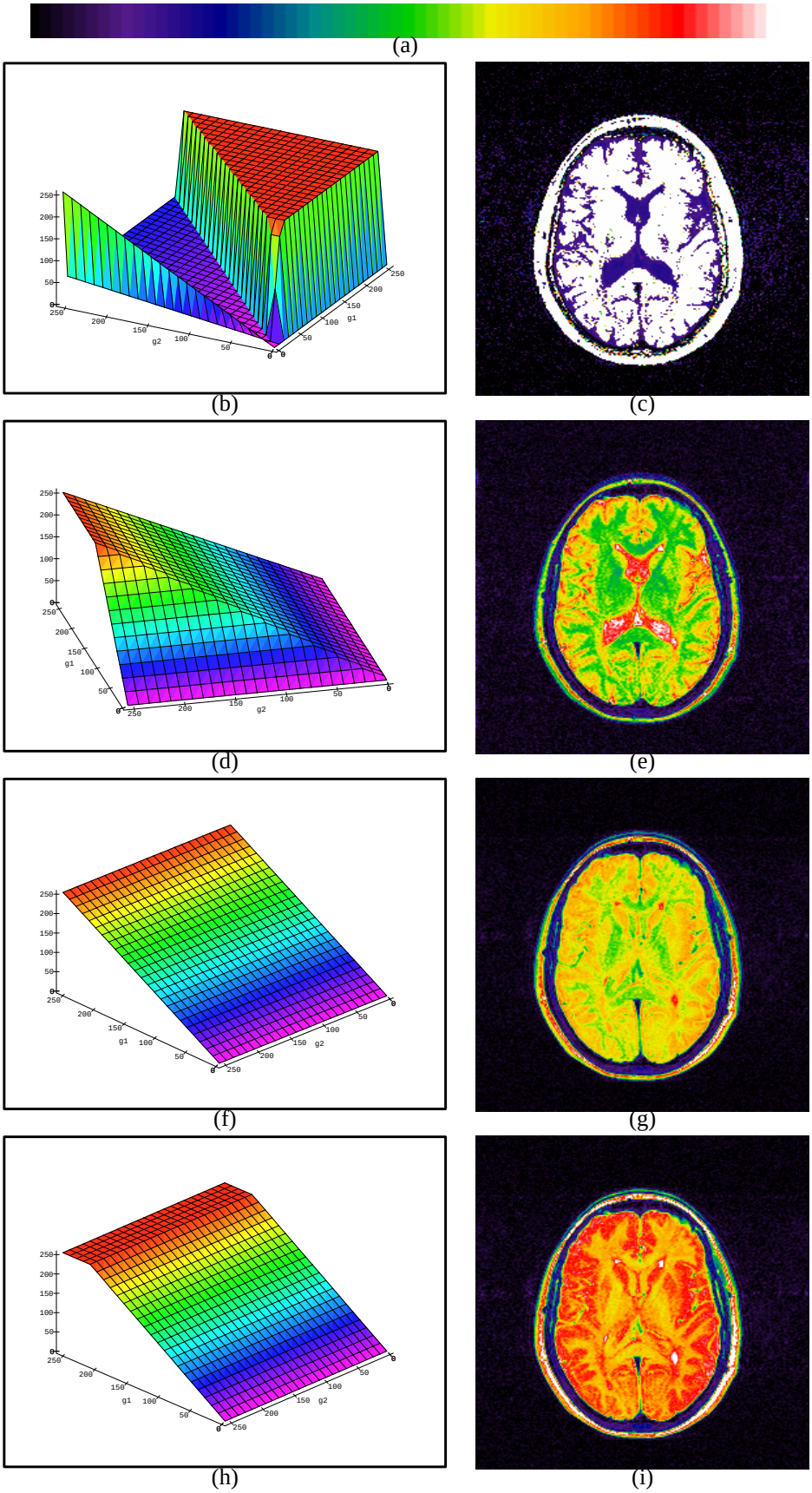

Figure 2: Experimental results on the GP-based interactive enhancement of MR images of the brain.

3.2 Interactive Evolution of Programs for Echocardiographic Image Enhancement

To test the ability of our method to produce general image enhancement algorithms for more noisy and less reproducible imaging techniques, in another set of experiments we used standard echographic images of the heart. The amount of noise and artifacts present in this kind of images can be easily understood by observing the original echocardiographic images in Figure 3.

The objective of these experiments was to evolve a pseudo-colouring algorithm capable of synthesising the motion of the heart into a single colour image. From the medical point of view this is a very important objective for several reasons: a) it is very difficult for the human visual system to evaluate the motility of the heart left ventricle (the heart chamber responsible for pumping blood into the aorta artery) only by looking at a still picture containing a sequence of frames, b) it is much easier to perceive and interpret colour, c) it is impractical to store in the patient record and handle video cassettes.

In order to reduce the complexity of the task, we decided to use only two video frames taken from sequences of 16 representing an entire heart beat of each patient: one representing the heart at the end of diastole (the phase in which the heart is completely relaxed and maximally expanded) and one at the end of the systole (when the heart is maximally contracted). The terminals `g1` and `g2` represent the gray levels of pixels in such frames, respectively.

In the experiments with echographic images we used two separate sets of images obtained from different patients: one for training (Figures 3(a) and (c)) and one for testing (Figures 3(b) and (d)). In the figure: (a) represents a section of the heart of a healthy subject at the end of diastole; (b) represents the heart of a different patient in diastole; (c) is the heart of the same patient as in (a) at the end of the systole; and (d) is the heart of the patient in (b) at the end of systole. (The gray levels of the images (a)–(d) have been reverted for displaying purposes.)

After a few warm-up runs, in a single run lasting 10 generations (i.e. involving 200 evaluations) we obtained the following two functions:

$$g1\,g2 - g2^2 + 2g2 - 3g1 - g2\,g1^2 + 0.854702 + \min(2 * g2 - 2 * g1, -g1 * (g1 - g2))$$

$$g1\,g2 - g2^2 + 2g2 - 2g1 - g2\,g1^2 + 0.648614 + g2 - g1$$

which have been simplified for an easier interpretation. The colour images produced by these functions are shown in Figures 3(e)–(h), both for the training images and the test images. Namely: (e) and (g) represent output produced by the two colouring algorithms on the training echocardiograms in Figures 3(a) and (c), while (f) and (h) are the output produced on the test echocardiograms in Figures 3(b) and (d).

It is worth noting that the behaviour of the colouring algorithms in the test images is very consistent with the behaviour on the trainining images. All the colouring algorithms evolved were judged very satisfactory by a cardiologist. These algorithms emphasise the motion of the heart walls by representing them with different colours in different phases: bright colours are used for systole, dark for diastole. However, they do so in a very non-linear way so as to give the doctor enough anatomical information to interpret the images.

3.3 Modelling the User

During the runs described in Section 3.1 we recorded the mean and variance of the output produced by each individual being compared by the user along with his choices. In this subsection we describe some preliminary results obtained with the approach for user-modelling proposed in Section 2.

In our experiments we used a fairly standard symbolic-regression-like strategy (population size=1,000, max generations=30, function set including arithmetic operators only, etc.) to build a model of the decisions made by the user during one run of the interactive evolution of programs for brain/ventricle enhancement. In such a run 100 fitness cases had been collected. GP easily evolved a program which was able to correctly predict 90 fitness cases out of 100.

Then we replaced the interactive selection-procedure with the user model induced by GP and ran GP with the same seed used to collect the data. As hoped, the results obtained by GP were very similar to those produced in the user-driven run. Although a few individuals in each generation (except the first) were different, the final result was hardly distinguishable from the one shown in Figure 2(c).

At that point we tried to assess the generalisation capabilities of the user model by running again the model-driven program while, at the same time, asking the user to choose the winner for each tournament. The input provided by the user was not passed to GP: it was only recorded to later compare the choices of the user with those of the model. The user was not aware of which image the model had chosen as the winner.

The results of this comparison were really surprising. In three separate runs with this setting the user and his model disagreed most of the times, namely 60, 55 and 52 times out of 100!

The explanation for this curious effect cannot simply be our using the wrong parameters to describe the phenotypes and therefore the choices made by the user. If that had been the case, we would have observed at least an agreement between model and user in about 50% of the cases, and also an inability of GP to learn the training examples. Despite the small number of tests, the difference between the estimated average disagreement, 56%, and 50% is statistically significant (assuming a 10% confidence coefficient and a Normal distribution).

The only explanation we have found so far for this effect is related to the fact that the user does tend to use quite a lot of global information on the composition of the population and on the history of choices previously made in the run. For example, we always tried to avoid premature convergence and,

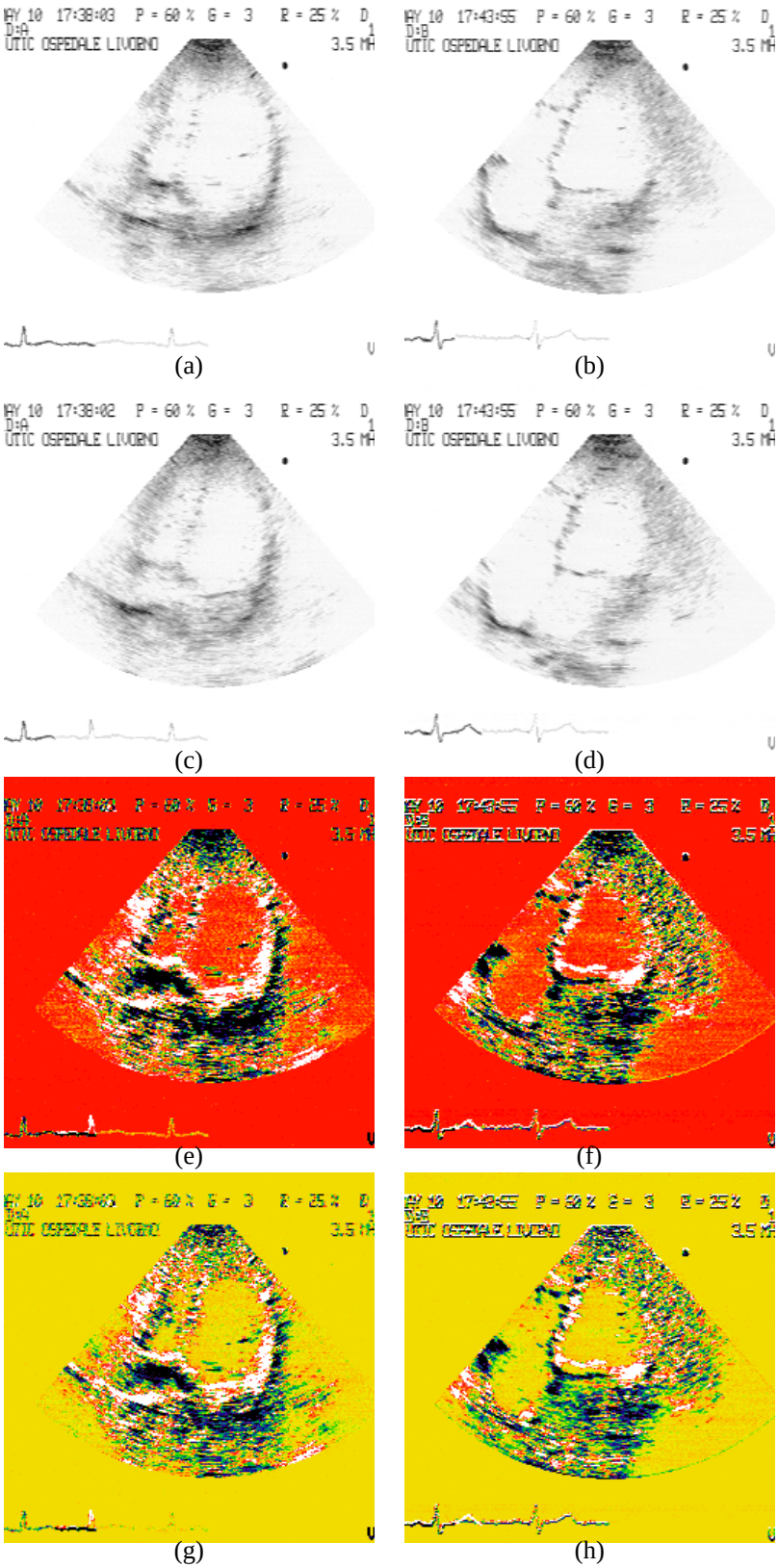

Figure 3: Original echocardiographic images used in our experiments (a)–(d) and output produced by two colouring algorithms evolved by GP (e)–(h) (see text for details).

sometimes instinctively sometimes on purpose, we tended to coevolve multiple phenotypical lines.

These and other behaviours of the user, which are probably crucial to obtain good solutions with so few evaluations, might vary so dramatically from run to run to be impossible to reproduce for a local, memory-less model. Further research will be needed to verify this conjecture.

4 Conclusions

In this paper we have presented an approach to the interactive development of programs for image enhancement with GP. In this approach the user drives GP by making decisions in tournament selection. The advantage of this technique is that no algorithmic fitness function or comparison criterion is needed and that the expertise and knowledge of the user can be used to direct evolution.

Confirming what has also been reported by others, our experiments have shown that the presence of the user in evolutionary algorithms somehow transforms them from a fairly inefficient search procedure, requiring hundreds of thousands of fitness evaluations, into extremely powerful and efficient methods which can produce good solutions to moderately hard problems in just hundreds of evaluations.

While this makes user-driven EAs very promising, the presence of the user within a computer-based system creates an unavoidable bottleneck due to the limited speed at which complex evaluations and comparisons can proceed. This bottleneck might seriously reduce the scalability of this class of methods.

To overcome the user bottleneck, we have also proposed a strategy based on the idea of automatically inducing programs modelling the user during interactive runs with small populations to be later deployed as user substitutes for larger-scale runs. Whether this will be a viable technique is something which remains to be established in future work, given the surprising results of our preliminary experiments.

Acknowledgements

The authors wish to thank W.B. Langdon and the other members of the EEBIC (Evolutionary and Emergent Behaviour Intelligence and Computation) group for useful discussions and comments. Dr. R. Del Bene of the Cardiologic Centre of the University of Florence and the Cardiology Department of the Hospital of Livorno are also thanked. This research is partially supported by a grant under the British Council-MURST/CRUI agreement.

References

[Andre, 1994] Andre, D. (1994). Automatically defined features: The simultaneous evolution of 2-dimensional feature detectors and an algorithm for using them. In Kinnear, Jr., K. E., editor, *Advances in Genetic Programming*, chapter 23, pages 477–494. MIT Press.

[Biles, 1994] Biles, J. A. (1994). GenJam: A genetic algorithm for generating jazz solos. In *Proceedings of the 1994 International Computer Music Conference, ICMA*, San Francisco.

[Cagnoni et al., 1991] Cagnoni, S., Caramella, D., Marin, E., and Valli, G. (1991). Computer assisted integration and display of diagnostic features in MR spin echo multi echo sequences. In Lemke, H., Rhodes, M., Jaffe, C., and Felix, R., editors, *Computer-Aided Radiology (Proc. CAR '91)*, pages 40–45, Berlin. Springer-Verlag.

[Daida et al., 1996] Daida, J. M., Hommes, J. D., Bersano-Begey, T. F., Ross, S. J., and Vesecky, J. F. (1996). Algorithm discovery using the genetic programming paradigm: Extracting low-contrast curvilinear features from SAR images of arctic ice. In Angeline, P. J. and Kinnear, Jr., K. E., editors, *Advances in Genetic Programming 2*, chapter 21, pages 417–442. MIT Press, Cambridge, MA, USA.

[Daida et al., 1995] Daida, J. M., Hommes, J. D., Ross, S. J., and Vesecky, J. F. (1995). Extracting curvilinear features from SAR images of arctic ice: Algorithm discovery using the genetic programming paradigm. In *Proceedings of IEEE International Geoscience and Remote Sensing, Florence, It*, volume 1, pages 673–675.

[Das et al., 1994] Das, S., Franguidakis, T., Papka, M., DeFanti, T. A., and Sandin, D. J. (1994). A genetic programming application in virtual reality. In *Proceedings of the first IEEE Conference on Evolutionary Computation*, volume 1, pages 480–484, Orlando, Florida, USA. IEEE Press. Part of 1994 IEEE World Congress on Computational Intelligence, Orlando, Florida.

[Dawkins, 1987] Dawkins, R. (1987). The evolution of evolvability. In *Proceedings of the Interdisciplinary Workshop on the Synthesis and Simulation of Living Systems (ALIFE '87)*, pages 201–220, Los Alamos, NM, USA.

[Fonseca and Fleming, 1995] Fonseca, C. and Fleming, P. (1995). An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation*, 3(1):1–17.

[Goldberg, 1989] Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, Massachusetts.

[Graf and Banzhaf, 1995] Graf, J. and Banzhaf, W. (1995). Expansion operator for interactive evolution. In *Proceedings of the IEEE Conference on Evolutionary Computation*, volume 2, pages 798–802. IEEE.

[Greenwood et al., 1996] Greenwood, G. W., Hu, X., and D'Ambrosio, J. G. (1996). Fitness functions for multipleobjective optimisation problems: Combining preferences with Pareto rankings. In Belew, R. K. and Vose, M., editors, *Foundations of Genetic Algorithms IV*, San Diego.

[Gruau, 1994] Gruau, F. (1994). *Neural Network Synthesis using Cellular Encoding and the Genetic Algorithm.* PhD thesis, Laboratoire de l'Informatique du Parallilisme, Ecole Normale Supirieure de Lyon, France.

[Gruau and Quatramaran, 1996] Gruau, F. and Quatramaran, K. (1996). Cellular encoding for interactive evolutionary robotics. Cognitive Science Research Paper 425, School of Cognitive and Computing Sciences, University of Sussex, Falmer, Brighton, Sussex, UK.

[Harris and Buxton, 1996] Harris, C. and Buxton, B. (1996). Evolving edge detectors with genetic programming. In Koza, J. R., Goldberg, D. E., Fogel, D. B., and Riolo, R. L., editors, *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 309–315, Stanford University, CA, USA. MIT Press.

[Johnson et al., 1994] Johnson, M. P., Maes, P., and Darrell, T. (1994). Evolving visual routines. In Brooks, R. A. and Maes, P., editors, *ARTIFICIAL LIFE IV, Proceedings of the fourth International Workshop on the Synthesis and Simulation of Living Systems*, pages 198–209, MIT, Cambridge, MA, USA. MIT Press.

[Koza, 1992] Koza, J. R. (1992). A genetic approach to the truck backer upper problem and the inter-twined spiral problem. In *Proceedings of IJCNN International Joint Conference on Neural Networks*, volume IV, pages 310–318. IEEE Press.

[Koza, 1994] Koza, J. R. (1994). *Genetic Programming II: Automatic Discovery of Reusable Programs.* MIT Pres, Cambridge, Massachusetts.

[Langdon, 1995] Langdon, W. B. (1995). Pareto, population partitioning, price and genetic programming. Research Note RN/95/29, University College London, Gower Street, London WC1E 6BT, UK.

[Poli, 1996a] Poli, R. (1996a). Genetic programming for feature detection and image segmentation. In Fogarty, T. C., editor, *Evolutionary Computing*, number 1143 in Lecture Notes in Computer Science, pages 110–125. Springer-Verlag, University of Sussex, UK.

[Poli, 1996b] Poli, R. (1996b). Genetic programming for image analysis. In Koza, J. R., Goldberg, D. E., Fogel, D. B., and Riolo, R. L., editors, *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 363–368, Stanford University, CA, USA. MIT Press.

[Sims, 1991] Sims, K. (1991). Artificial evolution for computer graphics. *ACM Computer Graphics*, 25(4):319–328. SIGGRAPH '91 Proceedings.

[Spector and Alpern, 1995] Spector, L. and Alpern, A. (1995). Induction and recapitulation of deep musical structure. In *Proceedings of International Joint Conference on Artificial Intelligence, IJCAI'95 Workshop on Music and AI*, Montreal, Quebec, Canada.

[Surry et al., 1995] Surry, P., Radcliffe, N., and Boyd, I. (1995). A multi-objective approach to constrained optimisation of gas supply networks: the COMOGA method. In Fogarty, T. C., editor, *Evolutionary Computing*, number 993 in Lecture Notes in Computer Science, pages 166–180, Sheffield, UK. Springer-Verlag.

[Tackett, 1993] Tackett, W. A. (1993). Genetic generation of "dendritic" trees for image classification. In *Proceedings of WCNN93*, pages IV 646–649. IEEE Press.

[Teller and Veloso, 1995] Teller, A. and Veloso, M. (1995). PADO: Learning tree structured algorithms for orchestration into an object recognition system. Technical Report CMU-CS-95-101, Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA.

[Webb, 1988] Webb, S. (1988). *The physics of medical imaging*. Institute of Physics Publishing. (Reprinted with corrections 1993).