On Two Approaches to Image Processing Algorithm Design for Binary Images using GP

Marcos I. Quintana[1], Riccardo Poli[2], and Ela Claridge[1]

[1] School of Computer Science, University of Birmingham, Birmingham, B15 2TT, UK. {M.I.Quintana, E.Claridge}@cs.bham.ac.uk
[2] Department of Computer Science, University of Essex, Colchester, CO4 3SQ, UK. RPoli@essex.ac.uk

Abstract. In this paper we describe and compare two different approaches to design image processing algorithms for binary images using Genetic Programming (GP). The first approach is based on the use of mathematical morphology primitives. The second is based on Sub-Machine-Code GP: a technique to speed up and extend GP based on the idea of exploiting the internal parallelism of sequential CPUs. In both cases the objective is to find programs which can transform binary images of a certain kind into other binary images containing just a particular characteristic of interest. In particular, here we focus on the extraction of three different features in music sheets.

1 Introduction

In the last few years, a variety of evolutionary approaches have been applied to the problem of discovering algorithms for image processing, so much so that Evolutionary Image Processing can almost be considered a proper field of research. A relatively small subset of these approaches (e.g. [1, 2, 3, 4, 5, 6]) have been based on Genetic Programming (GP) [7].

Mathematical Morphology (MM) is well known as a powerful tool for various image-processing tasks [8]. It is suitable for shape related processing since morphological operations are directly related to the object shape. To design a MM procedure (i.e. algorithm) some expert knowledge is necessary to properly select the structuring elements and to make an adequate selection of the morphological operators sequence. To date, no one has attempted to evolve MM algorithms using GP (although Yoda *et al.* [9] have suggested using genetic algorithms for this task). In this paper we start by describing the results of our efforts in this direction.

GP is usually quite demanding from the computation load and memory use point of view. This is particularly true when using GP to solve image processing problems, where the number of fitness cases can be extremely large, and finding improvements to the programs in a relatively small population can be difficult (which can easily lead to extensive bloat). Our approach based on GP with MM primitives is not immune from this. So, as a second contribution in this paper we have explored an alternative representation and primitive set which is based

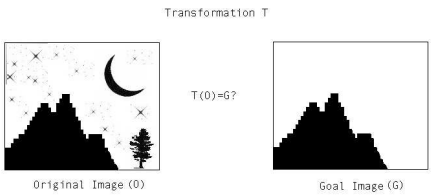

Fig. 1. Transformation from Original Image (O) to Goal Image (G).

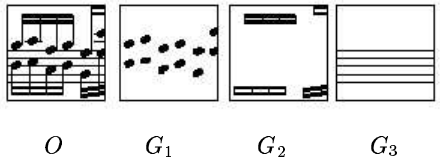

Fig. 2. Examples of binary transformations. The original fragment of a music sheet (O) and three possible goals extracted by hand: heads (G_1), hooks (G_2) and lines (G_3).

on a technique to speed up GP. Many ideas have been applied to improve the performance of GP. The technique we have used is known as Sub-Machine-Code GP (SMCGP) [10, 11]. SMCGP is a GP variant aimed to exploit the intrinsic parallelism of sequential CPUs. SMCGP extends the scope of GP to the evolution of parallel programs running on sequential computers. These programs are faster as, thanks to the parallelism of the CPU, they perform multiple calculations during a single program evaluation.

Adorni *et al.* have suggested using SMCGP for the efficient design of low-level vision algorithms [12, 13, 14]. They applied SMCGP in a plate detection and character recognition approach. The results obtained show that SMCGP is able to generate programs that are both accurate and efficient.

Here we have used GP (and SMCGP) to evolve transformation algorithms on music sheets. We aim to transform an original image into a goal image containing only a particular feature. The features analyzed are heads, hooks and lines.

The paper has the following structure. In Section 2 we describe the problem at hand and its features. In Section 3 we describe the GP approach for MM algorithm design on binary images. In Section 4 we describe how SMCGP can be used to speed up the evolution. In Section 5 we analyze the obtained results. Finally in Section 6 we draw some conclusions.

2 Test Problem and Fitness Function

Many image processing tasks for binary images can simply be described by providing an original binary image and a goal image (possibly drawn by hand). As shown in Figure 1 the problem then is to find an appropriate computational procedure which can perform that same transformation.

For the experiments presented in this paper we study the extraction of heads, hooks and lines in music sheets. The different features in a music sheet are interesting for the image transformation problem since each feature is easily recognizable for a human but the features involved are easy to miss-detect for a machine vision system. A set of images belonging to the training set is shown in Figure 2.

All our experiments were performed using a fitness function F ($0 \leq F \leq 1$) that evaluates the similarity between two images in a way that directs evolution towards a good trade-off between *sensitivity* (SV) and *specificity* (SP):

$$F = 1 - \frac{\sqrt{(1-SP)^2 + (1-SV)^2}}{\sqrt{2}}, \tag{1}$$

where $SV = \frac{TP}{TP+FN}$ and $SP = \frac{TN}{FP+TN}$, and TP is the number of true positives, FP is the number of false positives, TN is the number of true negatives and FN is the number of false negatives. For these experiments, the goal is to convert the original image (O) into the goal image (G). If $G(x_i, y_i) = 1$ then we have a true positive if $O(x_i, y_i) = 1$. If, instead, $O(x_i, y_i) = 0$ we have a false positive error. If $G(x_i, y_i) = 0$ then we have a true negative if $O(x_i, y_i) = 0$, and a false negative if $O(x_i, y_i) = 1$. The true/false positive/negative numbers are obtained by integrating over every pair of (x_i, y_i) coordinates in the original image.

3 GP for MM Algorithm Design

Let us start by describing our approach to evolving image processing algorithms based on GP and MM primitives.[3]

3.1 Basic Morphological Operations

The language of MM is *set theory*. As such, morphology offers a unified and powerful approach to numerous image-processing problems. Sets in MM represent the sets of objects in an image. For example, the set of all black pixels in a binary image is a complete description of the image. In binary images, the sets in question are members of the 2–D integer space Z^2, where each element of a set is a tuple (x, y) representing the coordinates of a black pixel in the image. Gray-scale digital images can be represented as sets whose components are in Z^3. Sets in higher dimensional spaces can contain other image attributes, such as color and time varying components.

The two morphological operations most used in MM are *dilation* and *erosion* and most of the algorithms developed by experts to perform a particular task make use of them.

If A and B are sets in Z^2 and $\emptyset$ denotes the empty set, the dilation of A by B, $A \oplus B$, requires performing the reflection of B about its origin, then shifting this reflection $\widehat{B}$ by x to obtain $(\widehat{B})_x$. The dilation of A by B, as shown in Equation 2, is the set of all x displacements such that $(\widehat{B})_x$ and A overlap by at least one nonzero element:

$$A \oplus B = \{x | (\widehat{B})_x \cap A \neq \emptyset\} \tag{2}$$

Figure 3 exemplifies this operation.

[3] Additional details on the experimental setup can be found in [15].

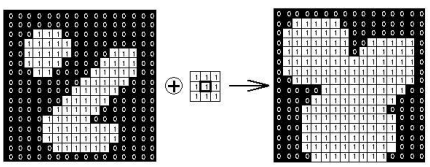

Fig. 3. Dilation.

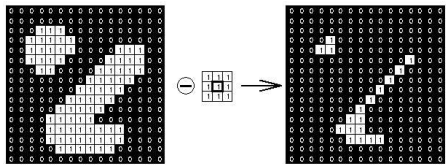

Fig. 4. Erosion.

The erosion of A by B, $A \ominus B$, is defined as the set of all points x such that $\widehat{B}$ translated by x is contained in A. The erosion of A by B, shown in Equation 3, is exemplified in Figure 4:

$$A \ominus B = \{x | (\widehat{B})_x \subseteq A\} \tag{3}$$

These two equations are not the only definitions for dilation and erosion, but they are usually preferred in practical implementations because of their analogy with the operation of convolution for linear filtering.

3.2 Evolution of Morphological Algorithms with GP

The GP approach suggested assumes that it is possible to a find a sequence of morphological operators in the MM algorithm's search space to convert an image into another containing only a particular feature of interest. To show this idea we select musical sheets as examples and extract some features from them.

The process is visualized in Figure 5. The first step is to create some examples of correct feature extraction by hand to be used as training sets in GP. Next, it is necessary to define the type, size and number of structuring elements to be used in the GP search. After the *primitive set* is defined, we start the GP search to obtain a (near) optimum tree representing a MM algorithm sequence (see below). The best sequence evolved, according to the fitness function, is also analyzed visually to decide whether its high fitness value corresponds to a good perceptual image quality.

When a MM algorithm is designed by hand, the programmer usually chooses a regular structuring element to make a sequence of operations. There is really no reason for choosing a regular element except that we, as humans, are more likely to understand *regularities* such as squares, lines, triangles, etc. However, it is entirely possible that one could get much better results by allowing *irregular* structuring elements. So, in our primitive set for GP we also include irregular structuring elements (selected randomly). We use structuring elements of sizes 3×3, 5×5 and 7×7 such as those shown in Figure 6. In this work, we include 11 regular and 11 irregular structuring elements of each size.

We use a function set including two functions, EVAL1 and EVAL2 of arity 1 and 2, respectively, which are used only for sequencing purposes. The terminal set includes nested primitives of the form $x(yz[w])$ where: $x \in \{e, d\}$, represents a morphological operators (erosion and dilation); $y \in \{R, I\}$, represents the type of

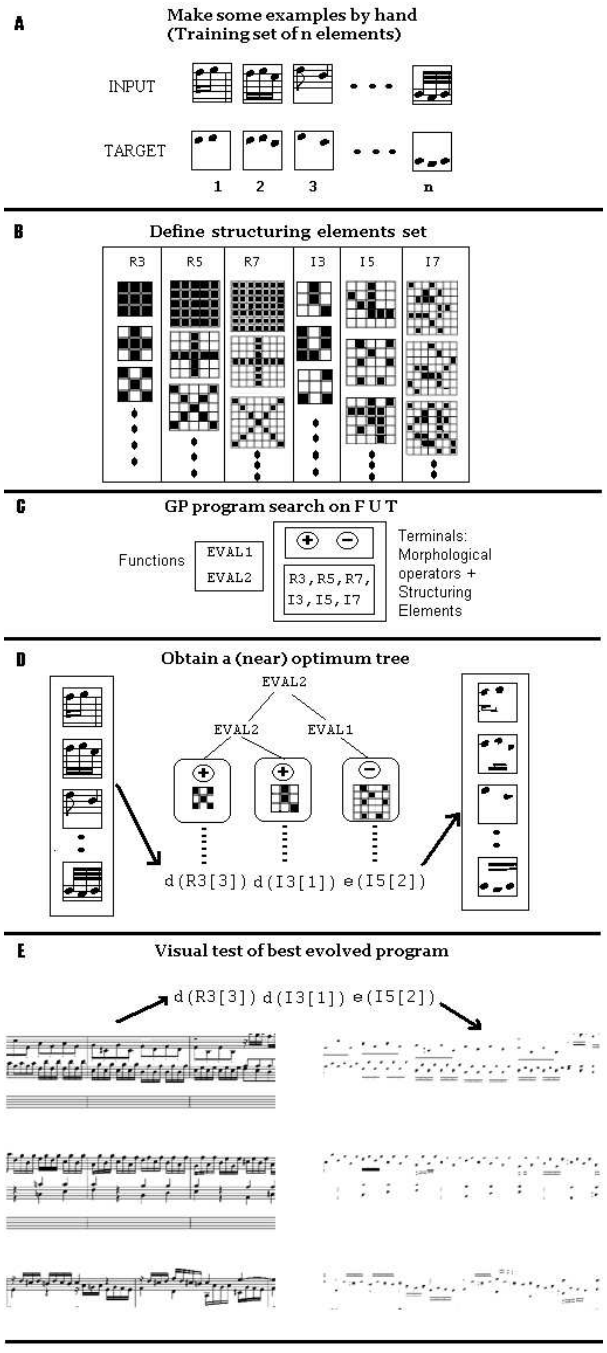

Fig. 5. Process to obtain a MM algorithm using GP. A) Make examples by hand. B) Decide structuring elements to use. C) Perform GP search. D) Obtain a (near) optimum tree. E) Evaluate best result.

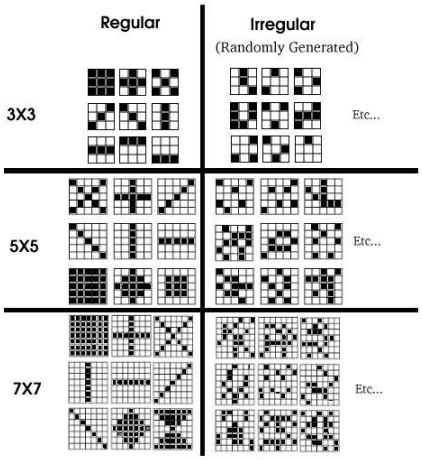

Fig. 6. Examples of regular and irregular structuring elements of various sizes.

structuring element selected (regular and irregular); $z \in \{3, 5, 7\}$, represents the size of structuring element and $w \in \{1..11\}$ represents the structuring element index.

As shown in Figure 5.D, at evaluation time each GP tree is transformed into a linear representation, which, when read from left to right, represents an MM algorithm. This is applied to the training set in order to evaluate the fitness of the corresponding GP tree.

The GP parameters we have used are similar to those suggested in the literature [7]. A crossover rate of 0.9 and 0.1 mutation rate were used. Mutation was based on the *ramped half and half* initialization method.

The sizes of the images in the training set were 16×16, 32×32, 64×64 and 128×128. The number of images in the training set was 1, 5, 15 and 25 in different experiments. Due to the high computation and memory load, all the experiments were performed using a population of 50 individuals run for 100 generations.

4 Using SMCGP to Evolve Image Processing Algorithms for Binary Images

In the MM approach described in Section 3 we were forced to use tiny populations and short runs, while, nowadays in GP usually people use much bigger populations and/or much longer runs.

So, we decided to test an alternative approach: we encoded the images using unsigned long integers to take advantage of the SMCGP paradigm. In this approach each image is represented as a vector of 32 unsigned long integers (32 bits); each element of the vector represents a row in the image. We used the

Table 1. Functions and terminals used in the SMCGP approach.

Functions	
AND	Bitwise AND
OR	Bitwise OR
NOT	Bitwise NOT
XOR	Bitwise XOR
SL	Bitwise shift left
SR	Bitwise shift right

Terminals	
X[1]	One binary image
X[2]	represented using 32
...	unsigned long integers
X[32]	(One row each.)

Function and Terminal sets presented in Table 1, where bitwise operations are applied to all the elements of a vector.

We used populations of 500, 1000 and 5000 programs run for 1000, 500 and 100 generations respectively (to fix the number to fitness evaluations to 500000 in each run) using 20 different random seeds. A 0.9 crossover rate was adopted. Mutation based on the ramped half and half initialization method was applied with a rate of 0.1.

5 Results and Analysis

5.1 GP with MM Approach

GP easily found different algorithms to solve the *heads* extraction problem by using MM primitives. The degree of accuracy is very difficult to evaluate visually and, depending on the evaluator, it does not always match the numerical fitness values obtained. However, we believe that many of the results obtained are as good as those that could be obtained by an expert writing the algorithm by hand. Figure 7 shows an original image and Figure 8 shows the result of an MM algorithm generated by GP.

The *hooks* extraction problem is a difficult task. That feature could be mismatched with lines, heads or other features present in a musical sheet. In spite of that, some GP generated algorithms present good approximations to the desired task. One example is presented in Figure 10.

Contrary to expectation, the *lines* extraction problem was the one presenting the most difficulties for the GP approach proposed. The results obtained were tending either to completely white images or to confuse lines with other features. We show an example in Figure 9 where GP accurately finds the lines, but also includes most of the hooks present in the test image.

5.2 SMCGP-based Approach

We made a total of 720 SMCGP runs. We had: 3 features to extract (heads, hooks and lines), 4 different numbers of examples in the training set (1, 5, 10,

15), 3 different population-size/number-of-generation combinations (500/1000, 1000/500 and 5000/100) and 20 different random seeds.

The SMCGP approach speeds up evolution in an impressive way. When the comparison is possible (recall that the GP and MM approach can't evolve so many programs) the SMCGP approach is up to 5 times faster. It is also able to evolve bigger populations for more generations. Indeed, the SMCGP approach finishes all the runs properly while some of the runs of GP with MM operators ran out of memory.

In terms of performance on the training set, the SMCGP approach appears to be as good as GP with MM. Due to space limitations we cannot provide examples of the output produced in this case.

6 Conclusions

We have described an approach to the evolution of image processing algorithms for binary images based on GP and MM. The approach has shown a good degree of accuracy in experiments with musical sheets. However, it has also shown a computational bottleneck when using big populations and running them for many generations.

As an alternative, we have explored a SMCGP-based approach which was hoped to speed up the evolution of algorithms for binary images. The SMCGP approach has been quite successful in this, making the evolution of big populations over a large number of generations possible. When a comparison is possible, SMCGP speeds up the evolution by 5 times w.r.t. the GP+MM approach, without any apparent loss in terms of performance.

Acknowledgments

The work presented in this paper was funded by the School of Computer Science at the University of Birmingham. Complementary funding was provided by ORS (UK) and SEP-Conacyt (Mexico). RP would like to thank the members of the NEC (Natural and Evolutionary Computation) group at Essex for helpful comments and discussion.

References

[1] Tackett, W.A.: Genetic programming for feature discovery and image discrimination. In Forrest, S., ed.: Proceedings of the 5th International Conference on Genetic Algorithms, ICGA-93, University of Illinois at Urbana-Champaign, Morgan Kaufmann (1993) 303–309

[2] Daida, J.M., Hommes, J.D., Ross, S.J., Vesecky, J.F.: Extracting curvilinear features from SAR images of arctic ice: Algorithm discovery using the genetic programming paradigm. In Stein, T., ed.: Proceedings of IEEE International Geoscience and Remote Sensing, Florence, Italy, IEEE Press (1995) 673–675

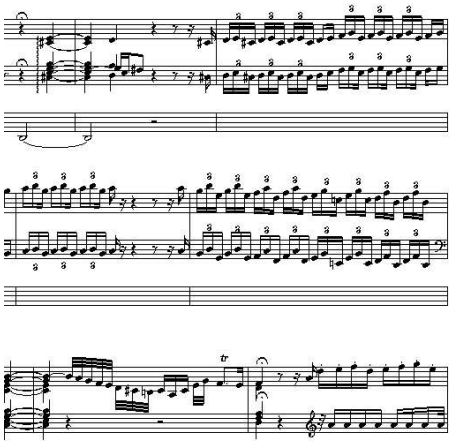

Fig. 7. Example of testing image for visual purposes.

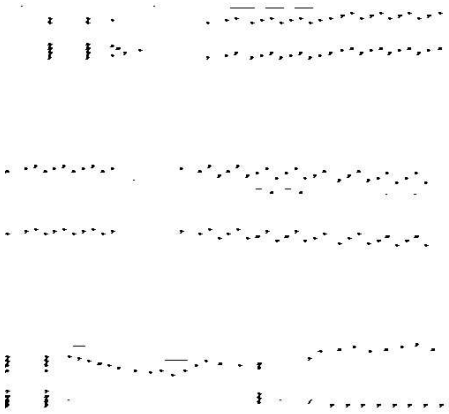

Fig. 8. Example of a good visual result for heads on the image in Figure 7 using GP and MM.

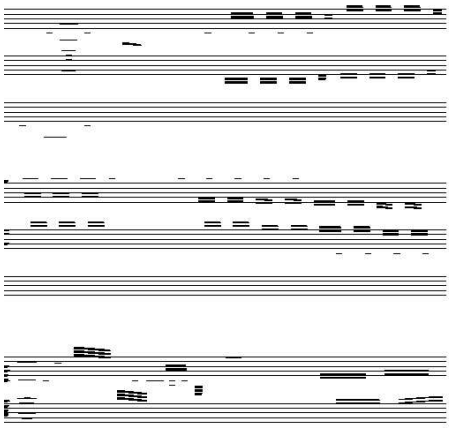

Fig. 9. Example of a good visual result for lines on the image in Figure 7 using GP and MM.

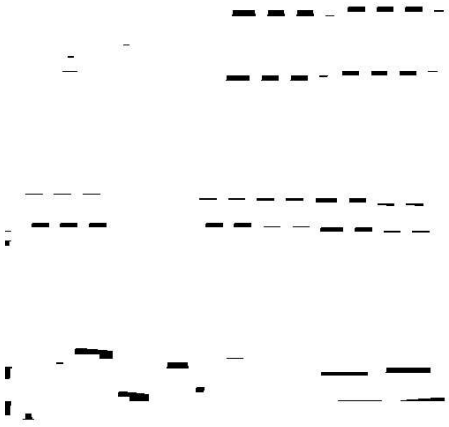

Fig. 10. Example of a good visual result for hooks on the image in Figure 7 using GP and MM..

[3] Poli, R.: Genetic programming for image analysis. In Koza, J.R., Goldberg, D.E., Fogel, D.B., Riolo, R.L., eds.: Genetic Programming 1996: Proceedings of the First Annual Conference, Stanford University, CA, USA, MIT Press (1996) 363–368
[4] Teller, A.: Evolving programmers: The co-evolution of intelligent recombination operators. In Angeline, P.J., Kinnear, Jr., K.E., eds.: Advances in Genetic Programming 2. MIT Press, Cambridge, MA, USA (1996) 45–68
[5] Howard, D., Roberts, S.C., Brankin, R.: Target detection in SAR imagery by genetic programming. In Koza, J.R., ed.: Late Breaking Papers at the Genetic Programming 1998 Conference, University of Wisconsin, Madison, Wisconsin, USA, Stanford University Bookstore (1998)
[6] Ebner, M., Zell, A.: Evolving a task specific image operator. In Poli, R., Voigt, H.M., Cagnoni, S., Corne, D., Smith, G.D., Fogarty, T.C., eds.: Evolutionary Image Analysis, Signal Processing and Telecommunications: First European Workshop, EvoIASP'99 and EuroEcTel'99. Volume 1596 of LNCS., Goteborg, Sweden, Springer-Verlag (1999) 74–89
[7] Koza, J.R.: Genetic programming: On the programming of computers by natural selection. MIT Press, Cambridge, Mass. (1992)
[8] Serra, J.: Image Analysis and Mathematical Morphology. Academic Press (1982)
[9] Yoda, I., Yamamoto, K., Yamada, H.: Automatic acquisition of hierarchical mathematical morphology procedures by genetic algorithms. Image and Vision Computing **17** (1999) 749–760
[10] Poli, R., Langdon, W.B.: Sub-machine-code genetic programming. In Spector, L., Langdon, W.B., O'Reilly, U.M., Angeline, P.J., eds.: Advances in Genetic Programming 3. MIT Press, Cambridge, MA, USA (1999) 301–323
[11] Poli, R.: Sub-machine-code GP: New results and extensions. In Poli, R., Nordin, P., Langdon, W.B., Fogarty, T.C., eds.: Genetic Programming, Proceedings of EuroGP'99. Volume 1598 of LNCS., Goteborg, Sweden, Springer-Verlag (1999) 65–82
[12] Adorni, G., Cagnoni, S., Gori, M., Mordonini, M.: Efficient low-resolution character recognition using sub-machine-code genetic programming. In: WILF 2001. (2002) In press.
[13] Adorni, G., Cagnoni, S., Mordonini, M.: Efficient low-level vision program design using sub-machine-code genetic programming. Workshop sulla Percezione e Visione nelle Macchine, available at citeseer.nj.nec.com/539182.html (2002)
[14] Adorni, G., Cagnoni, S.: Design of explicitly or implicitly parallel low-resolution character recognition algorithms by means of genetic programming. In R., R., M., K., Ovaska, S., Furuhashi, T., F., H., eds.: Soft Computing and Industry: Recent Applications. (Proc. 6th Online Conference on Soft Computing), Springer (2002) 387–398
[15] Quintana, M.I., Poli, R., Claridge, E.: Genetic programming for mathematical morphology algorithm design on binary images. In: KBCS-2002, International Conference on Knowledge Based Systems, KBCS 2002, NCST, Mumbai, India (2002) In press.